COMPUTERS & SECURITY 84 (2019) 318–333

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

A lightweight privacy and integrity preserving range query scheme for mobile cloud computing

Zhou Xu, Yaping Lin, Voundi Koe Arthur Sandor, Zhisheng Huang, Xinbo Liu*

College of Computer Science and Electronic Engineering, Hunan University, Changsha, China

ARTICLE INFO

Article history:
Received 13 August 2018
Revised 21 March 2019
Accepted 8 April 2019
Available online 15 April 2019

Keywords:
Mobile cloud computing
Privacy
Integrity
Range query
Lightweight

ABSTRACT

Mobile cloud computing has received a particular attention in recent years thanks to the expansion and easy accessibility of smartphones with enhanced capabilities, pushing for data owners to outsource more data to the cloud in order to serve the increasing number of mobile users. To protect data privacy, data have to be encrypted prior to outsourcing. Nevertheless, mobile users should still be able to query data, get the results corresponding to their requests and verify the integrity of the query results, all within a query privacy-preserving and efficient fashion. We focus in this work on the secure range query allowing to search for data belonging to a specific range. At the difference of previous works where data privacy and query results verification require separate structures and thus more overhead, our work achieves both data privacy and query results completeness verification using the encrypted data index and the vector neighbor chain (VNC). As a response to the intrinsic limitations of mobile devices, we design a lightweight protocol based on linear algebra operations for both 1-dimensional and multi-dimensional data. Practical experiments analysis on a real-world U.S. Census demographic dataset shows that our scheme guarantees the privacy-preserving property and is efficient in terms of computation and communication overhead.

1. Introduction

With the evolution of cloud computing, it is now possible to store data online and retrieve them through query services from any location, using any type of device, under a pay-per-use policy. Current data analysis from the Cisco visual networking index (VNI) report in CIS (2015) shows that the smartphone traffic will have increased by 49% in 2021, paving the way for more services of interest to be designed for mobile devices with respect to their inherent limitations as depicted in Abolfazli et al. (2015). Such services can be grouped under the term mobile cloud computing. We focus in this work on the service of mobile cloud data storage, where mobile devices access outsourced cloud data.

For privacy concerns, data owners are reluctant to outsource their sensitive data to the cloud service platform. A solution has been proposed in Kamara and Lauter (2010) as to encrypt sensitive data prior to their outsourcing. However encrypting data reduces their utility and downloading all the data before attempting to decrypt them locally will incur a large communication overhead, and due to the pay-as-you-go service model, huge financial expenditures. Searchable encryption has been proposed in Song et al. (2002) to perform search over encrypted data using keyword-based queries. In general, two types of query are used over encrypted

* Corresponding author.
E-mail addresses: xzhou@hnu.edu.cn (Z. Xu), yplin@hnu.edu.cn (Y. Lin), arthurvoundi@hnu.edu.cn (V.K. Arthur Sandor), zshuang@hnu.edu.cn (Z. Huang), xinbo_liu@hnu.edu.cn (X. Liu).
https://doi.org/10.1016/j.cose.2019.04.003
0167-4048/

data: range queries and aggregate queries as depicted in Samanthula et al. (2014). Although range queries have mostly been discussed in two-tiered wireless sensor networks as in Sheng and Li (2008); Zhang et al. (2009), and Tsou et al. (2013), we aim in this paper at applying secure range queries into the mobile cloud data storage environment.

1.1. Motivation

As most of the existing schemes performing range queries over encrypted data are oriented towards wireless sensor networks, we would like to apply the secure range query processing into mobile cloud storage. Additionally, previous works on secure range queries reveal the access patterns to the cloud server as in Samanthula and Jiang (2013) and exhibit high computational overhead with the use of expensive key cryptosystem operations. Query results integrity verification moreover is another point of interest as query results can be used as basis for critical decisions such as huge financial transactions or military actions as illustrated in Shi et al. (2009).

1.2. Technical challenges

There are three major challenges to address when performing secure range queries in mobile cloud data storage. (1):The first challenge is that cloud servers should not learn anything else from user queries, such as access patterns or search patterns, other than encrypting the data and the queries at its possession; (2):The second challenge is that data users should always be able to verify the integrity of query results; (3):The final technical challenge refers to the design of a secure range query scheme that presents overall minimal computing and storage and communication overhead for both the data owner and the data user.

1.3. Limitations of prior art

Existing works for range queries mainly include bucketing technique, order-preserving encryption method and homomorphic encryption technique. The bucketing technique aiming at partitioning the whole dataset into buckets of various sizes was proposed in Hacigms et al. (2002). However, based on the dataset knowledge and the historical query results, attackers could statistically estimate both queries and data items. The scheme moreover led to heavy communication overhead due to false positives in query results. An order-preserving encryption scheme (OPES) allowing the ciphertext to maintain the relative ordering of data in the corresponding plaintext was later proposed in Agrawal et al. (2004). However, as mentioned in Karras et al. (2016), this scheme also reveals the order in the data, hence is unattractive from the security viewpoint. The work in Tsou et al. (2013) proposed a scheme called EQ (Efficient Query) to solve the false positives issue as well as data integrity verification and data privacy through an order preserving encryption mechanism based on XOR linked list. It however reveals the access pattern and require large amount of shared secret information between a data owner and its data users. Fully homomorphic encryption (FHE) was proposed in Gentry (2009). However it exhibits high computation overload due to expensive public key cryptosystem operations and does not enable indexing of encrypted data.

To guarantee query result completeness verification, Shi et al. (2009) proposed a spatiotemporal crosscheck approach in order to reduce the communication overhead of previous bucketing schemes. However a compromised sensor can send fake data index to other sensors and to the storage node, thus compromising the integrity check. An excellent piece of work called SafeQ was proposed by Chen and Liu (2010), relying on a neighborhood chain data structure to verify query result completeness. However as each data item needs to be stored twice in the scheme, there is an overall increase in the amount of space and the communication overhead for both sensors and storage nodes, as well as an overhead in power consumption. The work in Yi et al. (2013) called QuerySec proposed a digital watermarking approach using a link watermarking to guarantee query results integrity. The proposed scheme required less space consumption than SafeQ. Chen et al. furthermore proposed in Hong et al. (2008) canonical range trees (CRT) to store counting information for multi-dimensional data which can be used for integrity verification. However, privacy preserving in CRT is a difficult problem and to the best of our knowledge no scheme has yet been proposed. Despite the various attempts to solve privacy and integrity in secure range queries, none of the above schemes has been designed with a mobile-oriented mindset.

To provide mobile devices friendly schemes with respect to their inherent limitations, the work in Karras et al. (2016) proposed a lightweight and indexable encryption based on linear algebra including vector and matrix operations. But the index structure in their scheme is not suitable for the range query in cloud environment. Li et al. (2018) proposed a scheme named LDSS (lightweight secure data sharing scheme) which outsources major computation overhead from the mobile devices to proxy servers. In Baharon et al. (2015) moreover, the authors proposed a new lightweight homomorphic encryption scheme for mobile cloud computing which incurs minimum computational overhead over encryption and key generation processes. Their scheme however exhibits a certain storage overhead due to the key management protocol.

1.4. Approach and results

Aiming at solving the above mentioned challenges, we present in this paper a lightweight encryption scheme which uses the linear algebra properties proposed in Karras et al. (2016) for data privacy protection. We however index encrypted data in our work which is different from the index construction for the query sequences in Karras et al. (2016), which adaptively indexes the data with regard to the continuously submitted user queries. For query result completeness verification, we propose a new chain technique denoted in our paper as vector neighbor chain(VNC) for both one and multi-dimensional data. In our work, the vector neighbor chain is directly embedded into the encryption redundancy information leading to a substantial minimum amount of overhead. Different from SafeQ, our scheme not only relies on the vector neighbor chain and the encrypted data index for data privacy, but also to verify the integrity of the query results. As outcome, our scheme

combines the encrypted data, encrypted index and the vector neighbor chain into a single structure, which does not generate additional storage and computation overhead.

To evaluate the performances of our proposed scheme, we perform extensive analysis on a real-world U.S. Census demographic dataset Kaggle obtained from Kaggle, both on one and two dimensional data respectively. Besides the analysis of security and integrity verification, we focus our experiment on the following metrics: the index construction time, the index size and the query processing time. Results analyses show that our scheme guarantees privacy and efficiency with respect to the mobile devices intrinsic limitations.

1.5. *Contributions*

The main contributions of this paper are as follows:

- We propose a lightweight and efficient scheme for preserving data privacy.
- We guarantee the query result completeness verification by solely relying on the vector neighbor chain and on the encrypted data index, without additional storage and computation overhead.
- We prove our scheme operations to be efficient in terms of computation and communication overhead for mobile devices.

1.6. *Paper organization*

The remainder of this paper is structured as follows. Section 2 reviews the related works and Section 3 represents our system models and problems. In Section 4,we introduce our privacy preserving scheme over 1-dimensional data in details and the integrity preserving scheme is represented in Section 5. In Section 6, we apply our scheme to multi-dimensional data. The complexity, security and integrity analyses are described in Section 7. The performance of our scheme is presented inSection 8. Finally, we conclude the paper in Section 9.

2. Related work

2.1. *Privacy preserving schemes*

Privacy preserving schemes allow data users to request data and to get corresponding results without leaking any information directly exploitable by an adversary. In order to guarantee the security and privacy of their sensitive uploaded data, data owners first encrypt them prior to outsourcing, as depicted in Kamara and Lauter (2010). To leverage encrypted data utility, researchers have come out with some general-purpose solutions making use of fully homomorphic encryption described in Gentry (2009) and oblivious RAM depicted in Goldreich and Ostrovsky (1996). However, as mentioned in Fan and Vercauteren (2012); Mani et al. (2013) and Boneh et al. (2005), due to their high computational cost, these methods are not efficient, especially as we are concerned with intrinsically resource-limited mobile devices. To improve on existing schemes, searchable encryption was introduced in Song et al. (2002) with great enhancements in terms of efficiency, functionality and security. Although searchable encryption focuses on keyword-based queries to retrieve data of interest, there are in general two types of queries over encrypted cloud data as stated in Samanthula et al. (2014): aggregate queries and range queries. Our work focuses on queries belonging to a certain range. Sheng and Li (2008) devised for the first time a privacy-preserving range query protocol for wireless sensor networks (WSN) which divided the domain of data into multiple disjoint buckets. This technique known as bucketing however incurs many false positives leading to a high communication cost. Another technique for range queries known as order preserving encryption scheme (OPES) was proposed in Agrawal et al. (2004). However OPES reveals the rank of the ciphertext, is vulnerable to statistical analysis, and requires considerable pre-shared volume of secret information between the data owner and its users. Chen et al. to reduce the high communication overhead of bucketing schemes introduced in Chen and Liu (2010) a scheme called SafeQ which uses a prefix membership verification technique developed in Cheng et al. (2007) to ensure data privacy. Despite being an excellent piece of work, the scheme still exhibits some computational and communicational overhead. The works in Zhang et al. (2014) and Li et al. (2014) use the bloom filter technique to process range queries. Furthermore, a scheme called PBtree proposed in Li et al. (2016) which achieves index indistinguishability by combining prefix encoding with bloom filter to perform secure range query. However due to the presence of false positives when relying on bloom filter method, the PBtree scheme cannot provide precise query results. Besides, the PBtree structure incurs a large space overhead. Even though the above mentioned schemes exhibit a considerable amount of overhead and are not suitable for mobile devices, we still rely on the work in Karras et al. (2016) which although incurring some computational overhead, makes use of interesting vectors and matrix linear algebraic operations to ensure data confidentiality.

There are also other papers focusing on different query privacy mechanisms. For example, in Li and Cao (2013), the authors addressed the problem of providing privacy aware incentives for mobile sensing, and proposed two privacy-aware incentive schemes to protect user privacy, where the query task consumes tokens and will be pre-computed. The works in Rios et al. (2017) provide a QPSP (Query Privacy for Sensing Platforms) protocol to protect the query privacy in Sensing-as-a-Service scenarios, where the query will be re-encrypted by the honest-but-curious sensing server after receiving the query from the user. In Zhang et al. (2017), a protocol called PPLQ (privacy-preserving proximity based location query) was proposed to allow a user to query the location of publishers using multi-dimensional search in the location queries. Although our focus is on range query, we however strongly bear in mind the challenges of designing a mobile-friendly scheme with respect to mobile devices' inherent limitations.

2.2. *Mobile computing range query*

Mobile cloud computing (MCC) as described in Abolfazli et al. (2015) utilizes cloud-based resources to provide resource augmentation to mobile devices through heterogeneous

wireless networks. Performing secure range queries in wireless sensors networks has come a long way but it is still in early stage when it comes to the MCC environment. To adapt to the inherent limitations of mobile devices, several lightweight schemes have been proposed in the literature. The work in Baharon et al. (2015) introduces a new lightweight homomorphic encryption (LHE) scheme for mobile cloud computing based on the work of Gentry (2009), with minimal computation power in both encryption and key generation stages. However as stated in Karras et al. (2016), homomorphic encryption does not allow encrypted data indexing. A lightweight and secure encryption protocol for the mobile cloud environment which relies on two unique data to seed the key was additionally proposed in Zegers et al. (2015). Moreover Li et al. (2018) proposed a lightweight and secure data sharing scheme called LDSS which migrates major computation overhead from the mobile devices onto proxy servers. One major concern with range queries in mobile cloud computing is that much more effort is put on functionality and less on security as it is not always of interest especially regarding functionality, to provide a trade-off between both. The work in Lin and Wu (2014) provides a directional continuous range query for mobile objects on road networks. However it does not operates on encrypted data. Obfuscation is mainly used in location-based services but it introduces much more communication overhead. We propose in this work to build an efficient and secure range query protocol for mobile cloud computing which provides a good trade-off between security, functionality and performances.

2.3. *Integrity preserving schemes*

To preserve the data integrity, a lot of work has been proposed in recent years. The merkle hash tree (MHT) technique was first proposed in Merkle (1980) for verifying the data integrity where the leaf of the MHT is the hash value of the data block, and the non-leaf node, the hash value of the string concatenation of its corresponding child nodes. The variants of MHT later proposed in Liu et al. (2010), Wu et al. (2015) and Zhang et al. (2012) for integrity verification of query results. In Pang et al. (2005), the signature is calculated by signing the digest of the concatenation between the tuple and its neighbors. The work in Narasimha and Tsudik (2006) uses signature aggregation and chaining to provide integrity verification over dynamic databases query results. The work in Chen and Liu (2010) denoted as SafeQ uses neighborhood chains to preserve the integrity of 1-dimensional and multi-dimensional data. However as each data item needs to be stored twice there is an overhead in storage, communication and power consumption which is not advantageous for the mobile cloud environment. Chen et al. furthermore proposed in Hong et al. (2008) canonical range trees (CRT) to store counting information for multi-dimensional data which can be used for integrity verification. However it is extremely difficult to guarantee privacy using CRT and to the best of our knowledge no such scheme has already been proposed. In Yi et al. (2013), the authors introduce a scheme called QuerySec which proposes a digital watermarking approach using a link watermarking to guarantee query results integrity for one dimensional data,and a data structure called multi-dimensional neighbor tree (MDNT) to validate

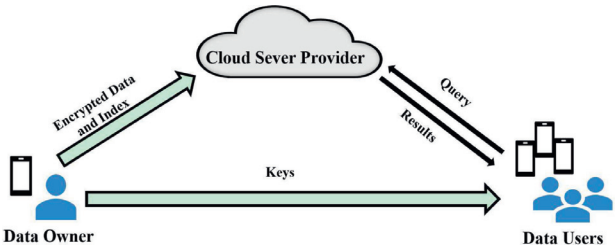


Fig. 1 – System model.

multi-dimensional data integrity. The proposed scheme requires less space consumption than SafeQ but still generates an overhead in energy consumption. Chen et al. moreover proposed in Fei and Liu (2014) a novel data structure called local bit matrices to detect damaged results with a high probability in cloud computing. However, the local bit matrices strongly depends on the data distribution to accurate examine the loss of data. The work in Yu et al. (2017) introduces a new data structure called S2L which achieves the integrity-preserving range query without incurring false positive. To provide query result integrity verification, our scheme builds a kd-tree, similar to the previous work in Kakde (2005), to index encrypted data. We however treat our kd-tree in a special manner along one splitting dimension which allows us to operate both query processing and query-result completeness verification without an additional structure. Further experiments on our scheme show that it provides privacy-preservation, integrity-preservation and efficiency towards the mobile cloud computing environment.

3. Problem formulation

3.1. *System model*

As shown in Fig. 1, our system model is made of three different entities: a data owner (DO), a cloud service provider (CSP) and multiple data users (DUs).

- The DO is an entity who encrypts his data and generates index over the encrypted data. Different from ordinary model, the DO in our model needs only to upload the encrypted index to the CSP, in which the encrypted data and the integrity verification information are all embedded. We provide more illustrations on this in further part of this paper.
- The DUs in our scheme are entities who are interested in getting access to some data that belongs to the specific range. The authorized DUs encrypt their query requests and send them to the CSP. The DUs decrypt the received encrypted data by using the decryption key obtained from DO.
- The role of the CSP is to provide storage services for the DO and to process the DUs query requests based on the index uploaded by DO, as well as, to return the results belong to the specific range to the DU who submitted the query.

We consider in this model that the DO and the different DUs are mobile users who should bear the cost of encryption and decryption operations with mobile devices.

3.2. Threat model

We consider in our work the DO and authorized DUs to be completely trusted entities. We follow the work in Ran et al. (1996) which considers the CSP to be honest-but-curious, meaning that the CSP in our model is a semi-trusted entity who strictly follows the required communication protocols as well as executes the required operations from the scheme, but may however be curious about the content of data and queries as even to deduce private content based on some background knowledge. A malicious CSP may attempt to speculate on the DO's private data and return undesirable query results that contain forged data, or more, attempt to exclude legitimate data. As the CSP itself is not trusted, there is a need in order to protect the data privacy, to design a secure but lightweight encryption scheme which constitutes a challenge for the mobile cloud environment, in terms of resources utilization given the inherent limitations of mobile devices. The cloud however should still be able to search data and to retrieve those matching the user-specified criteria with the help of the encrypted index.

We additionally consider in this model that the communication mean amongst entities is established through wireless connectivity. Compared to the traditional cloud computing system model using wired communication means, the wireless channel is more insecure and unstable. We however assume in this work that the communication channel is reliable and so the interactivity between DO and DUs is considered totally safe and secure.

3.3. Design goals

For range queries, what the cloud server needs to do is to determine which data falls within the given range. We encrypt both DO's data before outsourcing as well as encrypt queries issued by the DU as to preserve data and query privacy. Thus, the primary goal is to ask the CSP to perform search on encrypted data and to return correct results, without any side information disclosure about the original value. Second, for query efficiency consideration, an index for encrypted data is required. Third, we need to design a protocol that allows data users to verify the integrity of the query results. The integrity here has two meanings: on the one hand, query results should not contain any data items that do not meet the query requirements; while on the other hand, any data items that satisfy the query should not be excluded. The verification protocol should be able to detect the bad behaviors of cloud server. Last but not least, our scheme is designed to be lightweight, which means it provides less computation and smaller storage overhead to accommodate the characteristics of mobile devices.

4. Privacy for 1-dimensional data

In this section, we first detail the encryption method which is based on Karras et al. (2016). Then we introduce our index scheme which is definitely different from the one in Karras et al. (2016). We extend our scheme to multi-dimensional data, which will be introduced in Section 5.

Given a data set $d_1, d_2, \ldots, d_n$ and a range query $[b_1, b_2]$, any one of the data items and query bound can be denoted as d and b respectively. To determine if a data is within the certain range, the cloud server needs to compute whether $d_i \in [b_1, b_2](i = 1, 2, \ldots, n)$ or not. To implement this computation, one solution for the cloud is to compare the data with the upper and lower bounds of the range, that is, to conduct the comparison between d and b. Equivalently, the cloud should check whether

$$d - b > 0. \tag{1}$$

We introduce the knowledge of linear algebra:

$$\begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{pmatrix} = x_1 \cdot y_1 + x_2 \cdot y_2 + \cdots + x_n \cdot y_n. \tag{2}$$

In the case of n being 2, we convert the inequality to follow the form:

$$d - b = \begin{pmatrix} d \\ -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ b \end{pmatrix} > 0. \tag{3}$$

Our encryption scheme is based on Inequality (3). We use mathematical operations on vectors and matrices as the fuzzy operation which can effectively hide the real data values.

4.1. Encryption for 1-dimensional data

The encryption operation can be divided into two steps. The first step is called vectorization operation, which transforms the individual data into a l-dimensional vector. The second is matrix multiplication operation, which further blurs the vector where the value of each element is completely changed. A more detailed introduction is as follows.

4.1.1. Vectorization operation

Our encryption scheme aims to transform the individual data value into a vector form. For a better understanding, let's assume the length of the obtained vector to be 5, of which the position 2 and 3 are reserved for the original data value and the number -1 respectively. We denote the vector developed from d as $\vec{d}$. Then we have

$$\vec{d} = \begin{pmatrix} v_1 \\ d \\ -1 \\ v_2 \\ v_3 \end{pmatrix}, \tag{4}$$

where v_1, v_2, v_3 are three embedded numbers. As we can see, the three embedded numbers still can form a vector of length 3, which is represented as $\vec{O}_d$ of $\vec{d}$, for example, as in the Equality (4), $\vec{O}_d = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$. Similarly, we denote the vector originating

from b as $\vec{b}$, where the position 2 and 3 are reserved for the number 1 and the query bound respectively. Then we have:

$$\vec{b} = \begin{pmatrix} w_1 \\ 1 \\ b \\ w_2 \\ w_3 \end{pmatrix}, \tag{5}$$

where w_1, w_2, w_3 also are three embedded numbers. We denote the vector that consists of these three numbers as $\vec{O}_b = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$. So far, the dot product of the two vectors can be computed:

$$\vec{d} \cdot \vec{b} = \begin{pmatrix} v_1 \\ d \\ -1 \\ v_2 \\ v_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ 1 \\ b \\ w_2 \\ w_3 \end{pmatrix} = \vec{O}_d \cdot \vec{O}_b + d - b. \tag{6}$$

Therefore the difference of $d - b$ can be obtained exactly when $\vec{O}_d \cdot \vec{O}_b = 0$. In this case, $\vec{O}_d$ and $\vec{O}_b$ are orthogonal vectors that can be offset against each other in the scalar vector product. Theoretically speaking, as orthogonal vectors are infinite, we can be sure that every time we encrypt a data item or query bound, the choice of the orthogonal vector to be embedded is not the same. As query bound needs to be compared with other data items, we may wish to assume that a base vector of length $l - 2$ is given, all vectors $\vec{O}_b$ that are chosen to be embedded into $\vec{b}$ should be collinear to the base vector, while all vectors $\vec{O}_d$ that are selected to be embedded into $\vec{d}$ should be orthogonal to the base vector. We use $\vec{O}_g$ to represent the given base vector. So for any vector $\vec{O}_b$ collinear to $\vec{O}_g$, we can express it like this:

$$\vec{O}_b = \alpha \vec{O}_g, \tag{7}$$

where α is any randomly selected factor.

So far, a query bound can be encrypted into a vector by embedding the vector $\vec{O}_b$ which is acquired by randomly selecting a factor α based on the base vector $\vec{O}_g$. No matter what the factor α is, we always obtain the difference in the form of $d - b$. However, it is not enough to preserve the privacy of the data. An adversary who gets the couple of $\vec{d}$ and $\vec{b}$ can easily calculate the product $\vec{d} \cdot \vec{b}$, and get the difference $d - b$. That is to say, both the order and the exact difference of the encrypted data are disclosed.

Actually, for the DO and the DU, they do not really care about the exact difference, but more of the sign of $d - b (> 0, < 0, or = 0)$. Another operation should be adopted to solve this problem in the process of vectorization. For each data item, a random positive factor $\beta (\beta > 0)$ is selected to be multiplied by the vector $\vec{d}$. We can calculate the form $\vec{d} \cdot \vec{b}$, then get the difference of the new form $\beta(d - b)$ which will not change the sign of the result.

4.1.2. *Matrix multiplication operation*

As described above, the vectorization operation brings us the vector form of data items and query bound value. What is more, the comparison between encrypted data and encrypted query bound is realized by calculating the scalar product of their vector form in a relatively secure way. However, that is not secure enough in case an adversary knows the position where the βd and the $-\beta$ reside within the vector $\vec{d}$. In such a situation, the adversary only needs to do a simple mathematical division and he can get the original data value. We introduce a matrix multiplication operation to resist this type of attack effectively.

In this section, we introduce an invertible square matrix. The dimension of the square matrix is the same as the length of the vector obtained through vectorization operation. As mentioned above, the length of the vector is denoted as l, and M is represented as the invertible square matrix, while M^{-1} is its inverse, M^T is its transpose. We further express our encryption as follows:

$$\vec{E}_d(d) = M^{-1}\vec{d}, \tag{8}$$

$$\vec{E}_b(b) = M^T\vec{b}, \tag{9}$$

where $\vec{E}_d(\cdot)$ is the encryption function for data items, $\vec{E}_b(\cdot)$ is the encryption function for query bound. It should be noted that this is a process in which a vector is multiplied by a matrix. Especially, $\vec{E}_d(\cdot)$ and $\vec{E}_b(\cdot)$ are also vectors of length l. So we can calculate:

$$\begin{aligned} \vec{E}_d(d) \cdot \vec{E}_b(b) &= (M^{-1}\vec{d}) \cdot (M^T\vec{b}) \\ &= (M^T\vec{b})^T(M^{-1}\vec{d}) \\ &= ((\vec{b})^T M)(M^{-1}\vec{d}) \\ &= (\vec{b})^T\vec{d} \\ &= \vec{d} \cdot \vec{b} \\ &= \beta(d - b), \end{aligned}$$

where $(\vec{b})^T$ is the transpose of $\vec{b}$. After the matrix multiplication operation, the comparison between data item and query bound still can be correctly conducted. What is more, the original data value is completely obscured. An adversary who learns the position information of data value and the number -1 cannot deduce the exact original value without the invertible square matrix M which is the main decryption key.

4.2. *Indexing for 1-dimensional data*

It is important for the cloud to be able to compare encrypted data value and encrypted query bound. To promote the efficiency of processing range query, an AVL tree is employed to index the data. The AVL tree is essentially a binary search tree with balanced function. Our encryption scheme allows the cloud to conduct the comparison between the encrypted data value and encrypted query bound, knowing that two encrypted data values or two encrypted query bounds cannot be compared. As the indexing operation is implemented by data owner and not by the cloud server, an index for plaintext is acceptable at first. Then, for every tree nodes, we transform its key value to cipher, which is presented as $\vec{E}_d(\cdot)$. Since each node contains a ciphertext of the data item, an index without a separate dataset is sufficient to perform query processing for the CSP. Fig. 2 illustrates the structure of AVL tree. Note that

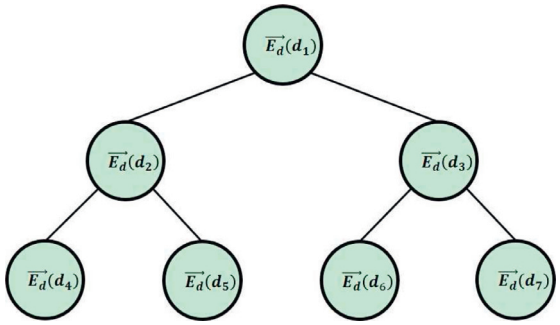


Fig. 2 – An AVL tree for encrypted data.

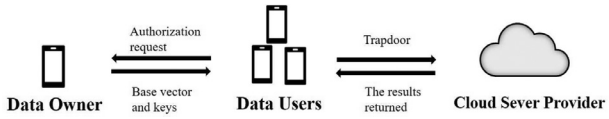


Fig. 3 – A complete query process diagram.

even the trusted client like the data owner actually know the value in the left node is smaller than the one in the right, the cloud always cannot compare the value of any two nodes. For example, if $d_1 > d_2$, the cloud has no way to get this relationship by calculating $\vec{E_d}(d_1) \cdot \vec{E_d}(d_2)$.

4.3. *Query processing for 1-dimensional data*

After the encryption operations, each query bound individually is encrypted into a l-dimensional vector form. The specific expression of the trapdoor for the range query $[b_1, b_2]$ can be denoted as $[\vec{E_b}(b_1).\vec{E_b}(b_2)]$. The trapdoor for a given range can be regarded as a $l \times 2$ matrix, of which each column corresponds to the vector form of a query bound. The DU who wants to retrieve data from the CSP computes the trapdoor of the range $[b_1, b_2]$ and sends it to the CSP. According to the index tree uploaded by the DO and the trapdoor computed by the DU, the CSP is able to search the data correctly. For a data item d and a range $[b_1, b_2]$, the CSP determines whether the data is in the range by calculating the formula $\vec{E_d}(d) \cdot \vec{E_b}(b_1)$ and $\vec{E_d}(d) \cdot \vec{E_b}(b_2)$. As we can see, $d \in [b_1, b_2]$ if and only if $\vec{E_d}(d) \cdot \vec{E_b}(b_1) > 0$ and $\vec{E_d}(d) \cdot \vec{E_b}(b_2) < 0$. Fig. 3 shows a complete range query process. The detailed steps are as follows:

(1) First of all, the DU indicates that he wants to retrieve data and applies for authorization from the DO;
(2) Then the DO sends the base vector and the decryption keys(includes decryption matrix and position information) to the authorized DU after confirming his identity;
(3) The authorized DU computes the trapdoor for the range query according to the given base vector, then sends the trapdoor to the CSP;
(4) After receiving a trapdoor computed by the DU, the CSP uses the trapdoor to search over the index tree and returns the required data to the DU;
(5) Finally, the DU decrypts the returned data using the decryption keys and obtain the original data.

5. Integrity for 1-dimensional data

To preserve the integrity of query result, the DU should verify whether the CSP has excluded some data items satisfying the query or forged some data items. We first propose a vector neighbor chain (VNC) which combines the encryption method described above to form a chain based on vectors. Given a data set $d_1, d_2, \ldots, d_n$, we define d_0 and d_{n+1} to represent the lower bound and the upper bound of the data set respectively. We first sort the data items in ascending order. We might as well assume that $d_0 < d_1 < d_2 < \ldots < d_n < d_{n+1}$. We use $L(d_j)$ and $R(d_j)$ to respectively denote the left and right neighboring value of d_j. When encrypting every individual data item into a vector form through the vectorization operation, we concatenate a data item with its left neighbor and right neighbor by inserting the neighboring values into the orthogonal vector. In Eq. (4), the vector form resulting from vectorization operation can be represented as $\begin{pmatrix} L(d) \\ d \\ -1 \\ R(d) \\ v_3 \end{pmatrix}$, where the original inserted numbers v_1 and v_2 in the orthogonal vector are replaced by the neighboring values of the data item d. For example, the vector form of the given data set $d_1, d_2, \ldots, d_n$ can be denoted as $\begin{pmatrix} d_0 \\ d_1 \\ -1 \\ d_2 \\ v_{3_1} \end{pmatrix}, \begin{pmatrix} d_1 \\ d_2 \\ -1 \\ d_3 \\ v_{3_2} \end{pmatrix}, \ldots, \begin{pmatrix} d_{n-1} \\ d_n \\ -1 \\ d_{n+1} \\ v_{3_n} \end{pmatrix}$, where each data value with its neighbors form a vector neighbor chain $\begin{pmatrix} d_0 \\ d_1 \\ d_2 \end{pmatrix}, \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}, \ldots, \begin{pmatrix} d_{n-1} \\ d_n \\ d_{n+1} \end{pmatrix}$. We argue that the position information about the data item and its neighbors is known to both DO and DU. The DO encrypts the data set with the inserted verification information, then uploads the encrypted data to the CSP.

The query response from CSP to the DU consists of two parts: the query result QR, which contains all data items that satisfy the query; and the verification object VO, which includes information for the DU to allow the integrity verification of QR. The verification object VO for 1-dimensional data is the right neighbor of the largest data item in QR. We argue that the premise of our algorithm is that the query processing follows a binary search. The method of finding the VO in an AVL tree is given by Algorithm 1.

We further detail the working mechanism for using the VNC to preserve the integrity of query result. We assume that $\vec{E_d}(d_i), \ldots, \vec{E_d}(d_j)(0 < i \le j \le n)$ are a set of QR for a query $[b_1, b_2]$ and $\vec{E_d}(d_{j+1})$ is the VO. After receiving the QR and the VO, the DU first decrypts them with the decryption matrix, then gets the vectors $\vec{d_i}, \ldots, \vec{d_j}$ and $\vec{d_{j+1}}$. Second, the DU extracts the data and the verification data that are represented as VNC, according to their position information. Finally, the integrity of QR is verified as follows:

- Case 1: If the CSP excludes the middle or the largest data item, the misbehavior can be detected because the VNC would be broken.
- Case 2: If the CSP excludes the smallest data item, say $\vec{E_d}(d_i)$, the existence of $\vec{E_d}(d_{i+1})$ is able to detect this

Algorithm 1 Finding *VO* for 1-dimensional data.

Input: The current node u of AVL tree, the query $[b_1, b_2]$
Output: *VO*

if ∃ a non-leaf node u that $u.KeyValue == b_2$ **then**
  if u has RightChild **then**
    $VO \leftarrow$ the leftmost node of $u.RightSubTree$
  else
    Consider u as a leaf node
  end if
else
  Find the leaf node t of the rightmost search path
  Let P denote the parent of t
  if $t.KeyValue \notin [b_1, b_2]$ **then**
    $VO \leftarrow t$
  else
    if $t == P.LeftChild$ **then**
      while $P.KeyValue \in [b_1, b_2]$ **do**
        $P \leftarrow P.Parent$
      end while
      $VO \leftarrow P$
    else
      $GP \leftarrow P.Parent$
      while $P == GP.RightChild$ && $GP \neq$ NULL **do**
        $P \leftarrow GP$
        $GP \leftarrow P.Parent$
      end while
      if $P == GP.LeftChild$ && $GP.KeyValue \notin [b_1, b_2]$ **then**
        $VO \leftarrow GP$
      end if
    end if
  end if
end if

Algorithm 2 Finding VO_2 for multi-dimensional data.

Input: The current node u of kd-tree, the z-dimensional query $[b_1^1, b_2^1], [b_1^2, b_2^2], \ldots, [b_1^z, b_2^z]$, the splitting dimension k
Output: VO_2

Let u^k denote the key value of u on the splitting dimension k
while $u^k \in [b_1^k, b_2^k]$ **do**
  The node u belongs to either QR or VO_2
  if ∃ a dimension $j (j \neq k)$ that $u^j \notin [b_1^j, b_2^j]$ **then**
    Return the node u as VO_2
  end if
end while

misbehavior because the $L(d_{i+1})$ in $\vec{d}_{i+1}$ is d_i that satisfies the query.

- Case 3: If the CSP returns an empty set as QR, the misbehavior can be detected because the VO $\vec{E}_d(d_{j+1})$ contains the value d_j represented as $L(d_{j+1})$ that satisfies the query.

Note that our VNC leverages the embedded data in the encryption process to support integrity verification, thus our verification technique does not require additional storage and computation overhead.

6. Queries over multi-dimensional data

A data set with multiple attributes is the most common used in practice. A z-dimensional data item D is a z-tuple $(d^1, \ldots, d^z)$ where $d^k (1 \leq k \leq z)$ is the data value in the k-th dimension. A multi-attribute data set can be denoted as $D_1, D_2, \ldots, D_n$, where $D_j = (d_j^1, \ldots, d_j^z) (1 \leq j \leq n)$. A range query for z-dimensional data is composed of z sub-queries where each sub-query over the data in the corresponding dimension is represented as $[b_1^k, b_2^k] (1 \leq k \leq z)$. If a z-dimensional data D is one of the results of a z-dimensional range query $[b_1^1, b_2^1], [b_1^2, b_2^2], \ldots, [b_1^z, b_2^z]$, then it means that the value in each dimension of the multi-dimensional data belongs to the query range of the corresponding dimension, i.e., $d^1 \in [b_1^1, b_2^1], d^2 \in [b_1^2, b_2^2], \ldots, d^z \in [b_1^z, b_2^z]$.

6.1. *Privacy for multi-dimensional data*

The encryption scheme based on vectors and matrices depicted above is now expanded onto multi-dimensional data. We adopt to the two-step encryption of 1-dimensional data into in the process of encrypting multi-dimensional data. After vectorization operation, each individual z-dimensional data item $D(d^1, \ldots, d^z)$ is transformed to a $z \times l$ matrix, where the meaning of l is the same as in the Section 4. For better comprehension, the value of l is still assumed to be 5. We denote the matrix developed from a data item D as M_D, the vectorization operation brings us a $z \times 5$ matrix as

$$M_D = \begin{pmatrix} v_{11}, & d^1, & -1, & v_{12}, & v_{13} \\ v_{21}, & d^2, & -1, & v_{22}, & v_{23} \\ \cdots, & \cdots, & \cdots, & \cdots, & \cdots \\ v_{z1}, & d^z, & -1, & v_{z2}, & v_{z3} \end{pmatrix}, \tag{10}$$

where $v_{ij} (i = 1, 2, \ldots, z; j = 1, 2, 3)$ is a set of embedded values that altogether make up a new $z \times 3$ matrix called O_D, i.e. $O_D = \begin{pmatrix} v_{11}, & v_{12}, & v_{13} \\ v_{21}, & v_{22}, & v_{23} \\ \cdots, & \cdots, & \cdots \\ v_{z1}, & v_{z2}, & v_{z3} \end{pmatrix}$. The same situation in the query bound. We denote the matrix developed from a sub-query $[b_1^k, b_2^k] (1 \leq k \leq z)$ as M_B, where each sub-query can be turned into a 5×2 matrix which is represented as

$$M_B = \begin{pmatrix} w_{11}, & w_{12} \\ 1, & 1 \\ b_1^k, & b_2^k \\ w_{21}, & w_{22} \\ w_{31}, & w_{32} \end{pmatrix}, \tag{11}$$

where $w_{ij} (i = 1, 2, 3; j = 1, 2)$ is also a set of embedded values that altogether make up a new 3×2 matrix called O_B, i.e. $O_B = \begin{pmatrix} w_{11}, & w_{12} \\ w_{21}, & w_{22} \\ w_{31}, & w_{32} \end{pmatrix}$. If two matrices O_D and O_B are multiplied to get a zero matrix, we can obtain the result represented as a $z \times 2$

matrix as a result of multiplying the two matrices M_D and M_B:

$$M_D M_B = \begin{pmatrix} d^1 - b_1^k, & d^1 - b_2^k \\ \cdots, & \cdots \\ d^k - b_1^k, & d^k - b_2^k \\ \cdots, & \cdots \\ d^z - b_1^k, & d^z - b_2^k \end{pmatrix} (1 < k < z). \quad (12)$$

In fact, in order to ensure that the two matrices can be multiplied to get a zero matrix, each row of matrix O_D is orthogonal to the corresponding column of matrix O_B, just as we did when encrypting 1-dimensional data. Each column of the matrix O_B is a different vector $\vec{O_b}$ expressed in Formula (7). As we can see, the difference between data value and query bound in the k-th dimension lies in the k-th row of the $z \times 2$ matrix while the difference that lies in the other rows has no practical significance as an interference item. For the process of vectorization operation, we select a random positive factor $\beta(\beta > 0)$ to multiply the matrix M_D. This operation is the same as the operation in 1-dimensional data.

The matrix multiplication operation further ensures data privacy as before. Unlike when dealing with 1-dimensional data, it is necessary to consider whether two matrices can be multiplied when dealing with multi-dimensional data. In order to properly deal with matrix multiplication, we perform some changes. We use the same invertible matrix M as when processing 1-dimensional data, with M^{-1} being its inverse. The matrix multiplication operation for multi-dimensional data can therefore be expressed as

$$E_D(D) = M_D M^{-1}, \quad (13)$$

$$E_B(B) = M M_B, \quad (14)$$

where $E_D(\cdot)$ is the encryption function for multi-dimensional data items, $E_B(\cdot)$ is the encryption function for each sub-query. $E_D(\cdot)$ should be a $z \times l$ matrix while $E_B(\cdot)$ is a $l \times 2$ matrix. So the multiplication of $E_D(D)$ and $E_B(B)$ can be conducted as

$$\begin{aligned} E_D(D)E_B(B) &= (M_D M^{-1})(M M_B) \\ &= M_D(M^{-1}M)M_B \\ &= M_D M_B. \end{aligned}$$

6.2. Indexing and query processing for multi-dimensional data

In order not to lose the generality, we use the the kd-tree data structure which has been first proposed to index multi-dimensional data for range query in Liu et al. (2010). However, in order to support integrity verification, the basis on which dimension we choose to divide is fixed from the beginning, then the midpoint of the fixed dimension is selected as the splitting value. Splitting on the midpoint will divide the data set into two subsets of roughly equal size, and the remaining subsets will be split recursively as well. The splitting dimension chosen by the DO will be sent to the authorized DU. Let us assume that a two-dimensional data set contains six two-dimensional data $(D_1, \ldots, D_6)$, we construct a kd-tree for this data set, as in Fig. 4. The parameter Split here represents the splitting dimension, which is decided by the DO, and is known to the DU.

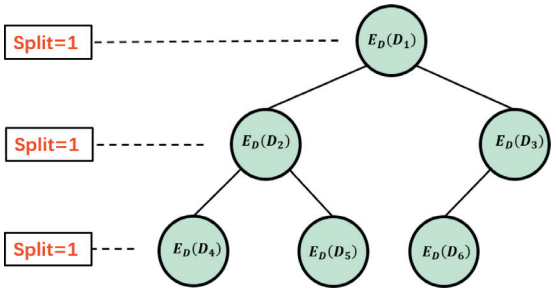


Fig. 4 – An example of kd-tree for two-dimensional data.

The process of multi-dimensional query processing is the same as for the 1-dimensional query processing given in Fig. 3. The detailed steps have already been introduced in the previous section, and will not be repeated here. A range query for z-dimensional data is composed of z sub-queries. For a z-dimensional data, it falls into a given range if and only if the data value in each dimension belongs to the sub-query range in the corresponding dimension. As the splitting dimension for index construction is known to the DU, the DU issues a multi-dimensional query to the CSP along with the splitting dimension. When processing the issued range query, the CSP retrieves the kd-tree nodes based on the splitting dimension. Only if the data value in the splitting dimension falls into the corresponding sub-query, the CSP will computationally decide whether the data value in other dimensions fall into the sub-queries of each of the corresponding dimension. If the data value in each dimension falls into the sub-query of the corresponding dimension, the CSP will return this multi-dimensional data as query result to the DU.

6.3. Integrity for multi-dimensional data

To preserve the integrity for multi-dimensional data, we use the VNC for multi-dimensional data built during the encryption process. In fact, given a z-dimensional data set $D_1, D_2, \ldots, D_n$, where $D_j = (d_j^1, \ldots, d_j^z)(1 \le j \le n)$, we use d_0^k and d_{n+1}^k to represent the lower bound and the upper bound of any data item along dimension k. We first sort the values on each dimension together with the lower bound and the upper bound in an ascending order. Then we use $L^k(d_j^k)$ and $R^k(d_j^k)$ to denote the left and right neighboring value of d_j^k along dimension k, respectively. When encrypting a z-dimensional data, we replace the corresponding two values for each row of O_D with the corresponding neighboring value. For instance, the formula (10) can be expressed as

$$M_D = \begin{pmatrix} L^1(d^1), & d^1, & -1, & R^1(d^1), & v_{13} \\ L^2(d^2), & d^2, & -1, & R^2(d^2), & v_{23} \\ \cdots, & \cdots, & \cdots, & \cdots, & \cdots \\ L^z(d^z), & d^z, & -1, & R^z(d^z), & v_{z3} \end{pmatrix}, \quad (15)$$

For each dimension $1 \le k \le z$, a VNC is formed along the dimension.

To verify the integrity of multi-dimensional data, the *VO* serving for verification is somewhat different from the 1-dimensional data. We argue that the *VO* for multi-dimensional data can be divided into two parts, namely VO_1 and VO_2. The first part, VO_1, is the same as in the 1-dimensional case, where the right neighbor of the largest data item along the splitting dimension that satisfies the corresponding sub-query is considered as *VO*. The second part, VO_2, is the set of data items whose value only on the splitting dimension satisfies the corresponding sub-query. Considering 6 examples of 2-dimensional data items (2,3), (4,7), (5,4), (7,2), (8,1), (9,6), we assume that the DO constructs index for the data set along the first dimension. According to the query protocol, for a range query ([4,8],[3,7]), $QR = (4, 7), (5, 4)$ with $VO_1 = (9, 6)$, and $VO_2 = (7, 2), (8, 1)$.

The items in $QR \cup VO_1 \cup VO_2$ should form a chain along the splitting dimension. The mechanism to verify the integrity for multi-dimensional data works just as in the 1-dimensional situation. The difference between both is how to find the *VO* for multi-dimensional data. The VO_1 can be found by executing the Algorithm 1 on the splitting dimension. According to our short introduction above VO_2, we provide an algorithm for finding VO_2 in the multi-dimensional case.

7. Complexity and security analysis

The encryption operation for data items is conducted by the data owner, while the computation for the trapdoor and the decryption operation are performed by the data user. Essentially, all these operations are achieved by mobile devices. Due to the limited resources of mobile devices, a lightweight and secure encryption scheme is required. In this section, the computational complexity, the security and integrity of our scheme are analyzed.

7.1. Complexity analysis

In our encryption process, the main calculation comes from the vectorization operation and the matrix multiplication operation. For 1-dimensional data, in order to encrypt a data item into a l-dimensional vector, l multiplications generated by the vectorization operation, l^2 multiplications and $l(l-1)$ additions generated by the matrix multiplication operation are required. For z-dimensional data, since every individual data is encrypted as a $z \times l$ matrix, zl multiplications during the vectorization operation, zl^2 multiplications and $zl(l-1)$ additions during the matrix multiplication operation are required. We assume that there are n unique data items included in the data set. As a result, it takes a total of $n(l^2+l)$ multiplications and $n(l^2-l)$ additions to encrypt the entire 1-dimensional data set, while computation for encrypting z-dimensional data is z times that of encrypting 1-dimensional data.

Similarly, a range query $[b_1, b_2]$ for 1-dimensional data can be considered as a 1-dimensional data set which contains only two data items, while a range query for z-dimensional data is composed of z 1-dimensional sub-queries. Therefore, $2(l^2+l)$ multiplications and $2(l^2-l)$ additions are totally required in total to encrypt an individual range.

As for the decryption process, there is a computation overhead only during the multiplication by the decryption matrix. Assume that the CSP returns r query results, for every individual result, it takes l^2 multiplications and l^2-l additions for 1-dimensional data, and zl^2 multiplications and $z(l^2-l)$ additions for z-dimensional data.

All calculations are summarized in Table 1 below.

7.2. Security analysis

In this section, we provide formal and informal security analysis of our scheme. Since our encryption scheme for multi-dimensional data is the extension of 1-dimensional case, we mainly analyze the 1-dimensional case for a better explanation. The encryption keys for data consist of the invertible matrix (including its inverse matrix and transposed matrix), the position information about original values, and the coefficient β used for data encryption.

7.2.1. Formal security analysis

We first define our encryption scheme as a (*Gen, Enc, Dec*) tuple where *Gen* is the key-generation algorithm, *Enc* is the encryption algorithm and *Dec* is the decryption algorithm. These algorithms in our scheme have the following characteristics:

–*Gen*: The *Gen* algorithm is a probabilistic algorithm which includes two pseudo-random generators. During the encryption phase, we use one of the pseudo-random generators to generate the parameter β introduced before and use another one to generate the invertible matrix.

–*Enc*: The encryption algorithm *Enc* takes as input the key k and a plaintext m, and outputs a ciphertext c. We denote the encryption as $Enc_k(m)$.

–*Dec*: The decryption algorithm *Dec* takes as input the key k and a ciphertext c, and outputs a message m. We denote the decryption as $Dec_k(c)$.

Next we consider the experiment of ciphertext-indistinguishability under chosen plaintext attack (IND-CPA) model proposed in [43]. The general IND-CPA game between a probabilistic, polynomial-time (PPT) adversary **A** and a challenger **C** is defined as follows.

1. The challenger **C** runs *Gen* to generate the key k.
2. The PPT adversary **A** outputs a pair of plaintext m_0, m_1 of the same length.
3. The challenger **C** randomly chooses a bit $b \in \{0, 1\}$ by flipping a coin and decides which plaintext to encrypt under the key k. The ciphertext is so called *challenge ciphertext* which is denoted as $c \leftarrow Enc_k(m_b)$.
4. The adversary **A** continues to have oracle access to $Enc_k(\cdot)$, and eventually outputs a bit b' as a guess.
5. If $b' = b$, we say the adversary **A** wins the game. Let $Pr[b' = b]$ be the probability that the adversary guesses correct, and let $Adv_{\mathbf{A}}^{IND-CPA}$ denote the advantage of **A** to win the game. An encryption scheme is secure under IND-CPA model if and only if the advantage of **A** to win the game is negligible, where $Adv_{\mathbf{A}}^{IND-CPA} = Pr[b' = b] - 1/2$.

Lemma: The proposed encryption scheme is ciphertext-indistinguishable under IND-CPA model with the PPT

Table 1 – Total required computations.

Operations	Implementer	Multiplications		Additions	
		1-d	z-d	1-d	z-d
Data encryption	DO	$n(l^2+l)$	$nz(l^2+l)$	$n(l^2-l)$	$nz(l^2-l)$
Trapdoor computation	DUs	$2(l^2+l)$	$2z(l^2+l)$	$2(l^2-l)$	$2z(l^2-l)$
Decryption	DUs	rl^2	zrl^2	$r(l^2-l)$	$zr(l^2-l)$

adversary's maximum advantage $\frac{2}{\eta \cdot l(l+3)(l-1)}$, where η is the range size of pseudo random numbers.

Proof. Now, for our proposed scheme, we reconsider the indistinguishability game. The game starts with a randomly generated key that includes a positive factor β and an invertible matrix with the dimension of $l \times l$ by running *Gen*, where l can be considered as a security parameter of our scheme. **A** outputs two plaintext data values d_0, d_1, then **C** randomly selects a bit i from {0, 1} and computes the *challenge ciphertext*, outputs a l-length vector $\vec{E_d}(d_i)$. The task for **A** is to determine which data is encrypted, with the continuous access to oracle $Enc_k(\cdot)$. Even if the adversary chooses the same plaintext to encrypt multiple times, the output of the encryption algorithm $Enc_k(\cdot)$ will generally be different each time. Furthermore, we assume that the adversary already has δ pairs of the original value and its ciphertext, l equations can be constructed for each such pair. Then the adversary can obtain at least $\frac{l^2+l-2}{l-1}+1 = l+3$ pairs of query bound value and its encrypted form. Eventually, the position information can be deduced by exhaustive search of $C_l^2 = \frac{l(l-1)}{2}$ ways for each pair of data ciphertext and query bound ciphertext with the probability of $\frac{2}{l(l+3)(l-1)}$. The ultimate goal for **A** is still to infer the invertible matrix based on known information. If the range size of pseudo random numbers is η, the probability that the adversary just guessed the correct parameter is $1/\eta$. Thus, we conclude that a PPT adversary can win the game with the maximum advantage of $\frac{2}{\eta \cdot l(l+3)(l-1)}$. □

7.2.2. *Informal security analysis*

In Karras et al. (2016), the authors have discussed the security for the encryption scheme in detail, we follow the basic idea and give the further analysis. As discussed above, the two-step fuzzy operation of vectorization and matrix multiplication makes the real data invisible. Assume that the attacker knows our encryption mechanism, we consider the following scenarios:

- Case 1: We consider the ciphertext attack which is the most common situation. First we discuss the data security for the vectorization operation. As described in Section 4, the two vectors $\vec{d}$ and $\vec{b}$ can be obtained from the original values d and b, respectively. An attacker thus can assume what the original values are and what the embedded numbers are. There are a total of $C_l^2 = \frac{l(l-1)}{2}$ assumptions he can make by selecting 2 out of l elements. In only one case, the product of two vectors consisting of embedded numbers is consistently equal to zero, which is the correct hypothesis. However, the security is further enhanced by performing the matrix multiplication operation. Just like the CSP, an attacker observes the ciphertext of data value and query bound. Then the attacker can only calculate the product of two ciphertext vectors such as $\vec{E_d}(d) \cdot \vec{E_b}(b)$, then get the sign of $d-b$, but not the exact difference of d and b. In this case, the attacker does not get any information of the original values, and even the order information of the data, because the two coefficients α and β are randomly selected each time during the encryption of data values and queries, which leads to the specific difference of d and b and the position information of the original values cannot be inferred. In general, we argue that our encryption scheme can effectively resist the ciphertext attack.
- Case 2: We furthermore consider in the case that an attacker could learn the position information of the values and the numbers 1 and -1 as described in Eqs. (4) and (5). After performing the matrix multiplication operation, the values in the vector are completely altered. Based on the first case we have discussed, the attacker knows the position information, so no longer need to make any assumptions to infer position of the original data. However, the values in the vector are completely altered after the matrix multiplication operation. The other $l-2$ elements reside in the ciphertext vector still cannot cancel each other anymore. The true value of data and query bound thus remain confidential.
- Case 3: We consider in the case that an attacker has access to construct a certain amount of ciphertext corresponding to the plaintext. The attacker can create a database for these plaintext-ciphertext pairs. Generally speaking, if the attacker later observes that a ciphertext is consistent with the items in his database, then he can get the corresponding plaintext. However, its impossible for our encryption scheme to produce the same ciphertext even if the same data is encrypted twice, because the orthogonal vector composed of the embedded numbers we randomly choose is unique for each encryption as described in Section 4.1. Therefore, the attacker can only attempt to derive the encryption key based on the plaintext-ciphertext pairs he has. For a pair of the original value and its ciphertext, l equations can be constructed. For example, we use $\begin{pmatrix} m_{11}, & m_{12}, & \cdots, & m_{1l} \\ m_{21}, & m_{22}, & \cdots, & m_{2l} \\ \cdots, & \cdots, & \cdots, & \cdots \\ m_{l1}, & m_{l2}, & \cdots, & m_{ll} \end{pmatrix}$ to denote the $l \times l$ inverse matrix M^{-1}, and use $\begin{pmatrix} v_1 \\ v_2 \\ \cdots \\ v_l \end{pmatrix}$ to denote the l-length vector $\vec{d}$ to be encrypted, where two out of the l elements are the original data value and the number -1 respectively. The corresponding ciphertext vector $\vec{E_d}(d)$ can be denoted as $\begin{pmatrix} E_1 \\ E_2 \\ \cdots \\ E_l \end{pmatrix}$.

For the attacker who has acquired a plaintext-ciphertext pair of the original value, according to Eq. (8), he can construct l equations as

$$\begin{cases} \beta(m_{11}v_1 + m_{12}v_2 + \cdots + m_{1l}v_l) = E_1 \\ \beta(m_{21}v_1 + m_{22}v_2 + \cdots + m_{2l}v_l) = E_2 \\ \cdots \\ \beta(m_{l1}v_1 + m_{l2}v_2 + \cdots + m_{ll}v_l) = E_l \end{cases} . \tag{16}$$

However, for the attacker, the l^2 elements from the matrix are unknown; the vector to be encrypted brings more $l-2$ unknowns; the positive factor is also unknown and is different for different plaintext-ciphertext pairs. In addition, the position information of the original values is still needed to be inferred. Obviously, the encryption key cannot be derived under such limited circumstances.

7.3. *Integrity analysis*

We propose a vector neighbor chain to verify the integrity of the query result. For integrity consideration, we analyze it in two aspects. On the one hand, the DU should verify whether there are data items in the query result that do not satisfy the query. A malicious CSP may tamper with some data. In our scheme, the matrix multiplication operation produces a completely ciphertext vector in which the value of each element is associated with a row or column of the matrix. As described in the previous paragraph, the ciphertext vector $\vec{E_d}(d)$ can be expressed as $\begin{pmatrix} m_{11}v_1+m_{12}v_2+\cdots+m_{1l}v_l \\ m_{21}v_1+m_{22}v_2+\cdots+m_{2l}v_l \\ \cdots \\ m_{l1}v_1+m_{l2}v_2+\cdots+m_{ll}v_l \end{pmatrix}$. Therefore, tampering with any value in the ciphertext vector will cause each value of the original vector to become different from the original value, thereby breaking the orthogonality for vectors consisting of embedded numbers, as well as the chain property for the entire vector. We argue that our scheme can effectively detect the misbehavior of tampering with data.

On the other hand, the DU should verify whether there are any data items that satisfy the query have been excluded. We first sort the data items in ascending order, then we concatenate each data item with its neighbors and form a vector. Note that we set the lower bound of the dataset as the left neighbor of the smallest data item, and set the upper bound as the largest data items right neighbor. When processing a range query, all data items that satisfied the query should be returned as QR. According to our query protocol, regardless of whether the QR contains data items, a VO containing at least one data item will be returned as verification. All data items in the QR together with the items in the VO should form a VNC. If a malicious CSP deletes some data items in the query result, the property of the chain is broken, so that the DU can easily detect the misbehavior.

8. Experiments and comparison analysis

In this section, we evaluate the performance of our scheme in terms of index construction time, index size and query processing time on a real-world U.S. Census demographic dataset (Kaggle) obtained from Kaggle. The dataset contains more than 70,000 data with more than 30 attributes, where a number of data records ranging from 10,000 to 70,000 with needed dimensionality are selected to conduct our experiments. We evaluate the performance of our scheme on one and two-dimensional data respectively. The experiments are conducted on a desktop PC running Windows 7 Ultimate with 4 GB RAM and a 3.4- GHz Intel Core i3-3240.

Table 2 – Parameter settings and security requirements.

Para.	Descriptions	Who Should Know		
		DO	DU	CSP
P	Position information about data and its neighbors	Yes	Yes	No
l	The size of key	Yes	Yes	Yes
$\vec{O_d}$	Vector used for data encryption	Yes	–	No
$\vec{O_g}$	Basic vector used for query bound encryption	Yes	Yes	No
α	Random factor multiplied by $\vec{O_g}$	–	Yes	No
β	Random positive factor	Yes	–	No
M	$l \times l$ invertible matrix(part of decryption key)	Yes	Yes	No

8.1. *Parameter setting*

Table 2. shows the key parameter settings and security requirements of our scheme. There are three entities included. The symbol '-' denotes that whether or not the entity knows the specific parameter, it has no effect on data security.

The parameter l represents the length of the vector in the case of one dimension and denotes one of the dimensions of the matrix in the case of two dimensions, as well as the size of keys. The changes in the value of l have an impact on the computational complexity and the security that have been analyzed in the previous section. For our experiments, we set the parameter l as 5, 10, 15 and 20, respectively.

The query result size is also a significant variable in our experiments. As is pointed in Kakde (2005), a range query on the kd-tree takes $O(n^{(1-1/z)} + r)$ time, where n is the total number of the data, z is the dimension and r is the number of reported points, i.e. the result size. The value of r is varies from 50 to 100 in our experiment.

8.2. *Evaluation on 1-dimensional data*

Firstly, we conducted our experiments on 1-dimensional data. Fig. 5 shows the experimental results in terms of index construction time and space consumption. As is shown in Fig. 5 (a), the average construction time of AVL tree for the length 5 ranges from 3.74 ms to 34.94 ms, for the length 10, it ranges from 6.94 ms to 67.60 ms, for the length 15 it is from 12.22 ms to 96.81 ms and for the length 20 it evolves from 16.78 ms to 130.86 ms, respectively. The average construction grows approximately linearly with the number of data items. Fig. 5 (b) shows that the index sizes range from 0.53 MB to 3.74 MB for the length 5. For the same number of data items, it takes 1.71 times for the length 10, 2.43 times for the length 15 and 3.14 times for the length 20 more than the case of length 5,

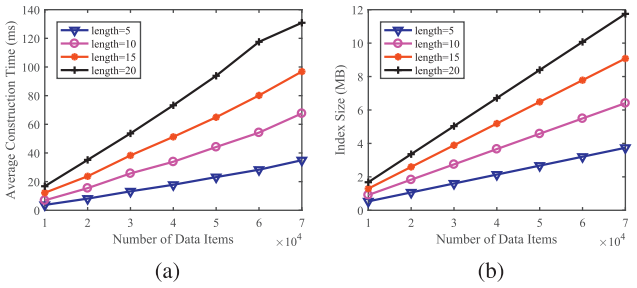


Fig. 5 – Time and size for 1-dimensional data indexing. (a) Average construction time. (b) Index size.

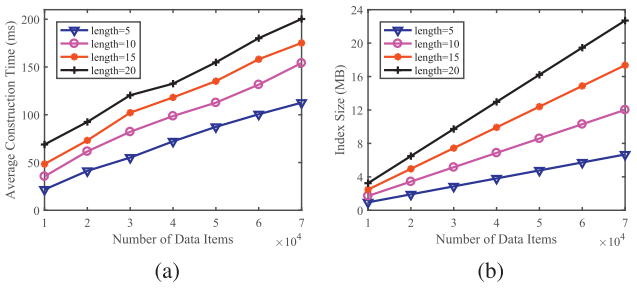


Fig. 7 – Time and size for 2-dimensional data indexing. (a) Average construction time. (b) Index size.

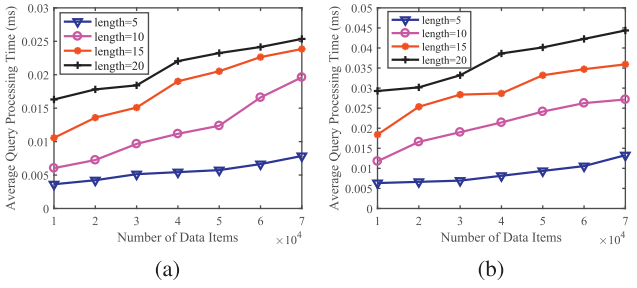


Fig. 6 – Average query processing time for 1-dimensional data. (a) *r*=50. (b) *r*=100.

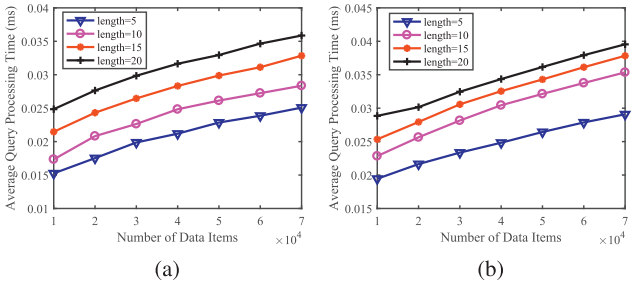


Fig. 8 – Average query processing time for 2-dimensional data. (a) *r*=50. (b) *r*=100.

respectively. The index construction is a one-time offline construction overhead in our experiments. For the length 20 and 70,000 data items, the space consumption is less than 12 MB.

In terms of query efficiency, we measured the average query processing time for different query result sizes varying from 50 to 100 on the experimental datasets. The experimental results are shown in Fig. 6, where the result size is denoted as r. We show the performance of two different result sizes of $r = 50$ and $r = 100$ for the different vector length here. For the same query result size, the average query processing time fluctuates within a certain range with the number of data items. For example, the average query processing time varies from 0.0036 ms to 0.078 ms for length 5 when the result size is fixed at 50. For the same the vector length, the average query processing time for $r = 50$ is 1.59 times, 1.77 times, 1.63 times and 1.75 times faster than the corresponding vector length for $r = 100$, respectively.

8.3. *Evaluation on 2-dimensional data*

We select two needed attributes of the U.S. Census demographic data set to form the 2-dimensional data set. The same performance indicators as the 1-dimensional experiments are evaluated in 2-dimensional experiments. The kd-tree construction time for 2-dimensional data set and the space consumption are shown in Fig. 7. Compared with 1-dimensional experiments, the trend of changes in the index average construction time and the space overhead of 2-dimensional experiments is the same as in the 1-dimensional experiments, but the overall cost of 2-dimensional experiments is greater. As is shown in Fig. 7 (a), the average construction time of kd-tree for the length 5 is 21.71 ms to 112.65 ms, while the time for the length 10, the length 15 and the length 20 is 1.38 times, 1.65 times and 1.93 times longer than the time for the length 5, respectively. We can observe in Fig. 7 (b) that it takes 22.70 MB space for the length 20 of 70,000 data items, which is 1.93 times more than the space consumption in 1-dimensional data.

The results of average query processing time for 2-dimensional data are shown in Fig. 8. Compared with 1-dimensional experiments, the query processing time of 2-dimensional data grows more smoothly. However, the query processing time of 2-dimensional data has a greater range of variation. For example, the average query processing time varies from 0.049 ms to 0.089 ms for length 20 when the result size is fixed at 100, while the time varies from 0.029 ms to 0.044 ms for 1-dimensional data. On the other hands, the change of the query time with the length is also smaller. As is shown in Fig. 8 (a), for the same the query result size of $r = 50$, the average query processing time for length 20 is just 1.76 times longer than the time for length 5, while the corresponding comparison in 1-dimensional experiments is 3.81 times. For the same the vector length, the average query processing time for $r = 50$ is 1.36 times, 1.35 times, 1.33 times and 1.37 times faster than the corresponding vector length for $r = 100$, respectively.

8.4. *Performance comparison*

To evaluate the performance of our scheme, we compare our index scheme with the PBtree index scheme (Li et al., 2016), which is an excellent solution for secure and fast range query. We consider the performance both for CSP-side and DO-side respectively. On the CSP-side, we focus on the efficiency of query processing. PBtree scheme employs a prefix coding and

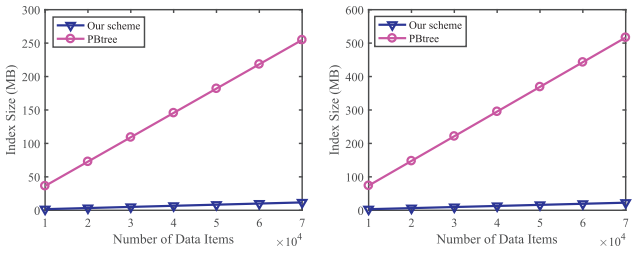


Fig. 9 – Index Size. (a) 1-dimensional data. (b) 2-dimensional data.

bloom filter to construct an indistinguishable index, as analyzed in Li et al. (2016), the worst-case search time complexity of PBtree is $O(r\log n)$, where r is the size of query result, n is the number of data items. In our scheme, we employ an AVL tree and a kd-tree for indexing 1-dimensional and 2-dimensional data, respectively. The worst-case search time complexity of AVL tree is given by $O(\log n)$, and the search time complexity of kd-tree for multi-dimensional ($n \geq 2$) data is given by $O(n^{(1-1/z)} + r)$ (Kakde, 2005), where z is the dimension of data. Within a certain amount of data, our scheme performs better than PBtree scheme in terms of query efficiency for 1-dimensional and 2-dimensional data. However the PBtree outperforms our scheme when the number of dimensions increases. Nevertheless, the query processing is conducted by the CSP which has enough computing resources.

We pay more attention to DO-side in which the owners are usually with the mobile devices having limited storage and processing capacity. We conduct an experiment to compare our scheme with PBtree scheme in terms of index overhead. We still use the U.S. Census demographic dataset Kaggle as mentioned before. In the experiment, on the one hand, we fix the parameter l for our scheme at 20, a larger size; and on the other hand, we use 18 bits to represent each data for PBtree scheme, as the values in the experimental dataset are all fall in the interval $[0, 2^{18}]$. The more details of the parameters setting can be found in Li et al. (2016). As shown in Fig. 9, the index size of PBtree scheme respectively grows from 36.40 MB to 254.82 MB for 1-dimensional data and from 73.83 MB to 516.83 MB for 2-dimensional data with the number of data varying from 10 thousand to 70 thousand, while the corresponding index size of our scheme increases from 1.68 MB to 11.75 MB and from 3.24 MB to 22.70 MB. The experimental results show that the storage space consumption of our scheme is much less than PBtree scheme. This also means the communication overhead of our scheme is much less than that of PBtree, since the data owner needs to send less data to CSP.

8.5. *Result summary*

In summary, the experimental analysis shows that our scheme enables secure range query in a much more efficient manner. For mobile devices that perform encryption, the operation is just a simple computation on vector and matrix, which means a minimal computation overhead. For query efficiency consideration, we employ a balanced binary tree to index the encrypted data. Since the nodes of the tree contain all the encrypted data and verification information, the data owner needs only to upload the index tree when outsourcing the data to the cloud without extra overhead. For decryption operation, the decryption keys consist of only a $l \times l$ matrix and of the position information of original values. The experimental results demonstrate the superiority of our scheme.

9. Conclusion

In this paper, we propose a lightweight encryption scheme for the mobile cloud computing environment, focusing on secure range query for numerical data. For 1-dimensional data, we employ an AVL tree for indexing, in order to improve the efficiency of the query processing. We additionally encrypt each individual data item into a vector form as well as perform some operations based on the matrix properties to ensure that the real data and their order are not disclosed. We further expand the encryption scheme used for 1-dimensional data onto the multi-dimensional data scenario, and use a kd-tree to index the data items for query efficiency consideration. Furthermore, we propose a vector neighbor chain to verify the completeness of query result without additional storage and computation overhead. Extensive experiments are conducted on the U.S. Census demographic dataset to evaluate the performance of our scheme and the experimental results show that it takes less than 100 ms for 2-dimensional data to construct index for 10 thousand data items for a vector of length 20. At the same time, the storage for index is completely affordable. For a 20-length vector, the index size for 70 thousand data items is less than 24 MB. The experimental results illustrate the superiority of our scheme.

Conflict of interest

- None of the authors of this paper has a financial or personal relationship with other people or organizations that could inappropriately influence or bias the content of the paper.
- It is to specifically state that "No Competing interests are at stake and there is No Conflict of Interest" with other people or organizations that could inappropriately influence or bias the content of the paper.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Project nos. 61872131, 61702180).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cose.2019.04.003.

REFERENCES

Abolfazli S, Sanaei Z, Sanaei MH, Shojafar M, Gani A. Mobile cloud computing: the-state-of-the-art, challenges, and future research. USA: Willeys & Sons; 2015.

Agrawal R, Kiernan J, Srikant R, Xu Y. Order preserving encryption for numeric data. In: International conference on management of data; 2004. p. 563–74.

Baharon MR, Shi Q, Llewellynjones D. A new lightweight homomorphic encryption scheme for mobile cloud computing. In: IEEE international conference on computer and information technology; Ubiquitous computing and communications; Dependable; 2015. p. 618–25.

Boneh D, Goh E, Nissim K. Evaluating 2-DNF formulas on Ciphertexts. In: Theory of cryptography conference; 2005. p. 325–41.

Chen F, Liu AX. SafeQ: secure and efficient query processing in sensor networks. In: Conference on information communications; 2010. p. 2642–50.

Cheng J, Yang H, Wong SHY, Zerfos P, Lu S. Design and implementation of cross-domain cooperative firewall. In: IEEE international conference on network protocols; 2007. p. 284–93.

Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption. IACR Cryptol 2012;2012:144. Eprint Archive

Fei C, Liu AX. Privacy and integrity preserving multi-dimensional range queries for cloud computing. Proceedings of the networking conference, 2014.

Gentry C. Fully homomorphic encryption using ideal lattices. In: Symposium on the theory of computing; 2009. p. 169–78.

Goldreich O, Ostrovsky R. Software protection and simulation on oblivious rams. J ACM 1996;43(3):431–73.

Hacigms H, Iyer BR, Chen L, Mehrotra S. Executing SQL over encrypted data in the database-service-provider model. Proceedings of the ACM SIGMOD international conference on management of data, 2002.

Hong C, Ma X, Hsu W, Li N, Wang Q. Access control friendly query verification for outsourced data publishing. Proceedings of the European symposium on computer security-ESORICS, 2008.

Kaggle. https://www.kaggle.com/muonneutrino/us-census-demographic-data.

Kakde HM. Range searching using KD tree, 2005.

Kamara S, Lauter K. Cryptographic cloud storage. In: Proceedings of the international conference on financial cryptograpy and data security; 2010. p. 136–49.

Karras P, Nikitin A, Saad M, Bhatt R, Antyukhov D, Idreos S. Adaptive indexing over encrypted numeric data. In: ACM international conference on management of data; 2016. p. 171–83.

Li G, Guo L, Gao X, Liao M. Bloom filter based processing algorithms for the multi-dimensional event query in wireless sensor networks. J Netw Comput Appl 2014;37:323–33.

Li Q, Cao G. Providing privacy-aware incentives for mobile sensing. Proceedings of the IEEE international conference on pervasive computing and communications; 2013. p. 76–84.

Li R, Liu AX, Wang AL, Bruhadeshwar B. Fast and scalable range query processing with strong privacy protection for cloud computing. IEEE ACM Trans Netw 2016;24(4):2305–18.

Li R, Shen C, He H, Gu X, Xu Z, Xu C. A lightweight secure data sharing scheme for mobile cloud computing. Proceedings of the IEEE international conference on cloud computing technology and science, 6; 2018. p. 344–57. 2

Lin C, Wu S. Processing directional continuous range queries for mobile objects on road networks. In: IEEE international conference on cyber technology in automation control and intelligent systems; 2014. p. 330–5.

Liu Y, Ning P, Dai H. Authenticating primary users' signals in cognitive radio networks via integrated cryptographic and wireless link signatures. Proceedings of the IEEE symposium on security and privacy; 2010. p. 286–301.

Mani M, Shah K, Gunda M. Enabling secure database as a service using fully homomorphic encryption: challenges and opportunities, 2013. arXiv: Databases

Merkle RC. In: Proceedings of the IEEE symposium on security and privacy. Protocols for public key cryptosystems; 1980. 122–122

Narasimha M, Tsudik G. Authentication of outsourced databases using signature aggregation and chaining. In: International conference on database systems for advanced applications; 2006. p. 420–36.

Pang HH, Jain A, Ramamritham K, Tan K. Verifying completeness of relational query results in data publishing. In: ACM Sigmod international conference on management of data; 2005. p. 407–18.

Ran C, Feige U, Goldreich O, Naor M. Adaptively secure multi-party computation. Proceedings of the annual ACM symposium on theory of computing; 1996. p. 639–48.

Rios R, Nunez D, Lopez J. Query privacy in sensing-as-a-service platforms. In: Information security; 2017. p. 141–54.

Samanthula BK, Jiang W. Efficient privacy-preserving range queries over encrypted data in cloud computing. Proceedings of the IEEE sixth international conference on cloud computing, 2013.

Samanthula BK, Jiang W, Bertino E. Privacy-preserving complex query evaluation over semantically secure encrypted data. European symposium on research in computer security; 2014. p. 400–18.

Sheng B, Li Q. Verifiable privacy-preserving range query in two-tiered sensor networks. In: Infocom the conference on computer communications IEEE; 2008. p. 46–50.

Shi J, Zhang R, Zhang Y. Secure range queries in tiered sensor networks. In: Infocom; 2009. p. 945–53.

Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. Proceedings of the IEEE symposium on security and privacy, 2002.

The zettabyte era: trends and analysis. Cisco, 2015. https://www.cisco.com/c/en/us/solutions/collateral/serviceprovider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.

Tsou YT, Lu CS, Kuo SY. Privacy- and integrity-preserving range query in wireless sensor networks. Proceedings of the global communications conference, 2013.

Wu D, Choi B, Xu J, Jensen CS. Authentication of moving top-k spatial keyword queries. IEEE Trans Knowl Data Eng 2015;27(4):922–35.

Yi Y, Rui L, Fei C, Liu AX, Lin Y. A digital watermarking approach to secure and precise range query processing in sensor networks. Proceedings of the Infocom, IEEE, 2013.

Yu L, Yu J, Lei W, Kuang L. Yquery: a novel privacy- and integrity-preserving range queries in two-tiered sensor networks. Proceedings of the international conference on electrical and information technologies for rail transportation, 2017.

Zegers W, Chang SY, Park Y, Gao JZ. A lightweight encryption and secure protocol for smartphone cloud. In: Service-oriented system engineering; 2015. p. 259–66.

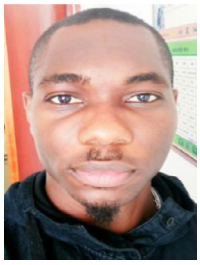
Zhang R, Shi J, Zhang Y. Secure multidimensional range queries in sensor networks. In: Acm Interational Symposium on Mobile Ad Hoc Networking and Computing; 2009. p. 197–206.

Zhang R, Zhang Y, Zhang C. Secure top-k query processing via untrusted location-based service providers. In: IEEE Infocom; 2012. p. 1170–8.

Zhang S, Lin Y, Liu Q, Jiang J, Yin B, Choo KR. Secure hitch in location based social networks. Comput Commun 2017;100:65–77.
Zhang X, Dong L, Peng H, Chen H, Li D, Li C. Achieving efficient and secure range query in two-tiered wireless sensor networks. In: Quality of Service; 2014. p. 380–8.

Zhou Xu received the B.S. degree in communication engineering from Hunan University, China, in 2016, where he is currently pursuing the M.S. degree in software engineering. His research interests include security and privacy issues in cloud and big data.

Yaping Lin received the B.S. degree in computer application from Hunan University, China, in 1982, the M.S. degree in computer application from the National University of Defense Technology, China, in 1985, and the Ph.D. degree in control theory and application from Hunan University in 2000. He has been a Professor and a Ph.D.Supervisor with Hunan University since 1996. From 2004 to 2005, he was a Visiting Researcher with the University of Texas at Arlington. His research interests include machine learning, network security, and wireless sensor networks.



Voundi Koe Arthur Sandor received his B.S in fundamental computer sciences from University of Yaoun0de I, Cameroon, in 2011, and his M.S degree in computer and application technology from Hunan University, China, in 2015. Since 2015, he has been a Ph.D. candidate in Network and Information Security at Hunan University, China. His research interests include network security, machine learning, and security and privacy issues in cloud.

Zhisheng Huang received the B.S. degree in software engineering from Hunan University, China, in 2016, where he is currently pursuing the M.S. degree in software engineering. His research interests include security and privacy issues in location-based services and big data.

Xinbo Liu received his M.S. degree in Analytical Science from Central South University, China, in 2015. Since 2015, he has been a Ph.D. candidate in College of Computer Science and Electronic Engineering, Hunan University. His research interests include cloud computing, network security, data mining and machine learning.