Secure k-NN Query on Encrypted Cloud Data with Multiple Keys

Ke Cheng, Liangmin Wang, *Member, IEEE,* Yulong Shen, Hua Wang, Yongzhi Wang, Xiaohong Jiang, *Senior Member, IEEE,* and Hong Zhong

Abstract—The k-nearest neighbors (k-NN) query is a fundamental primitive in spatial and multimedia databases. It has extensive applications in location-based services, classification & clustering and so on. With the promise of confidentiality and privacy, massive data are increasingly outsourced to cloud in the encrypted form for enjoying the advantages of cloud computing (e.g., reduce storage and query processing costs). Recently, many schemes have been proposed to support k-NN query on encrypted cloud data. However, prior works have all assumed that the query users (QUs) are fully-trusted and know the key of the data owner (DO), which is used to encrypt and decrypt outsourced data. The assumptions are unrealistic in many situations, since many users are neither trusted nor knowing the key. In this paper, we propose a novel scheme for secure k-NN query on encrypted cloud data with multiple keys, in which the DO and each QU all hold their own different keys, and do not share them with each other; meanwhile, the DO encrypts and decrypts outsourced data using the key of his own. Our scheme is constructed by a distributed two trapdoors public-key cryptosystem (DT-PKC) and a set of protocols of secure two-party computation, which not only preserves the data confidentiality and query privacy but also supports the offline data owner. Our extensive theoretical and experimental evaluations demonstrate the effectiveness of our scheme in terms of security and performance.

Index Terms—Data security, k-NN query, multiple keys, cloud computing.

1 Introduction

Recently, cloud computing has become an increasingly popular service for its flexibility and scalability, which motivates many organizations, institutions and companies to prefer to outsource data services to cloud platform [1]. At the same time, much attention has been paid to cope with the special security and privacy problems in outsourced cloud [2, 3]. On one hand, to protect the data confidentiality, the data owner (DO) encrypt the sensitive information of his outsourced data, such as income level, health records, personal photos before the dataset is uploaded to the cloud [4, 5]. On the other hand, data owner may plan to rely on cloud platform for querying of the datasets stored in cloud, not just for storage and management. Therefore, a large amount of secure schemes have been proposed [6, 7] to support the query over encrypted cloud data.

As a fundamental query operation in spatial and multimedia databases, k-nearest neighbors (k-NN) query aims at identifying k nearest points for a given query point in a dataset. In the past few years, researchers have proposed various methods to address the security and privacy problems of k-NN query on encrypted cloud data. The general approach is to encrypt data by the data owner (DO) before outsourcing; the authorized query users (QUs) perform a complex series of encryption and decryption operations during query execution. For example, the work in [8] proposes an asymmetric scalar-product-preserving encryption (ASPE) to preserve scalar product between the query vector and any vector for distance comparison, which is sufficient to find k-NN. Instead of finding exact nearest neighbor, Yao et al. [9] allow a cloud party to approximate it based on secure Voronoi diagram algorithm. Elmehdwi et al. [10] propose a novel protocol over encrypted data based on a Twin-Cloud [11] model and Paillier cryptosystem [12], which can calculate k-NN between data records and query records in a secure manner. However, all the above schemes have assumed that the query users are fully-trusted and have the access to the key for encrypting and decrypting outsourced data. It will bring about several problems in the real world. Firstly, cloud platform can totally break the outsourced database once the key is obtained from any compromised query user. It is obvious that each query user could be one of the lucrative targets for attackers. Secondly, data owner may have no enough trust on each query user in many applications which will limit the scope of these schemes. For instance, hospitals or institutes of medicine might contribute medical data for a disease classification study or a service available to doctors [13, 14]. Thus, doctors

- *K. Cheng is with the School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China, 710071, and the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China, 230601. E-mail:chengke@ahu.edu.cn*
- *L. Wang is with the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China, 230601. E-mail:wanglm@ujs.edu.cn*
- *Y. Shen is with the School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China, 710071. E-mail:ylshen@mail.xidian.edu.cn*
- *H. Wang is with the Centre of Applied Informatics, College of Engineering and Science, Victoria University, Australia/Department of Computer Science, Taiyuan Normal University, China. Email:hua.wang@vu.edu.au*
- *Y. Wang is with the School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China, 710071. E-mail:yzwang@xidian.edu.cn*
- *X. Jiang is with the School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China, 710071, and the School of Systems Information Science, Future University Hakodate, 116-2, Kameda Nakano-Cho, Hakodate, Hokkaido, 041-8655, Japan. E-mail:jiang@fun.ac.jp*
- *H. Zhong is with the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China, 230601. E-mail:zhongh@ahu.edu.cn.*

Manuscript received April 19, 2005; revised August 26, 2015.

can search the k-NN cases with some similar physiological data to help treat patients. If using the above schemes, the doctors will encrypt the indices with the same key as the one that the data owner encrypts and decrypts the outsourced database. Obviously, it is not realistic, since the data owner do not want to release the medical data in the clear to each other or a cloud platform. Thirdly, once query users receive the key, their query processing will not be controlled by data owner any more, and it is difficult to revoke the access even they are deemed to be untrustworthy. In general, these schemes with key-sharing are still far from being practical in most instances.

To solve the above mentioned problem, the work in [15] improves the ASPE scheme in [8] to solve the problem without sharing key with query users. Instead, query users interact with the data owner to derive a query encryption. That is, these schemes require data owner to be constantly online. Zhu et al. [16] lately propose an improvement of the scheme in [15] ,which can support the offline data owner. However, these schemes disclose more or less information about data owner's key. More importantly, the kernel of the works [15, 16] is the ASPE method, but the ASPE method cannot be proved to resist the chosen-plaintext attack (CPA) [9]. The work in [17] also gives a secure NN (k = 1) query scheme which can resist the collusion between cloud server and query users. Nevertheless, the work is not practical because of the strong assumption.

In this paper, considering the above problems, we focus on the secure k-NN query over encrypted cloud data without key-sharing. Firstly, based on the distributed two trapdoors public-key cryptosystem (DT-PKC) [18], we construct a set of protocols of secure two-party computation that will be used as sub-routines of our proposed scheme. Furthermore, we propose a novel secure k-NN scheme with multiple keys to address the above problems. Specifically, in our scheme, each query user holds his own unrelated key and the data owner can encrypt and decrypt outsourced data using the key of his own, without sharing the key with the query users. Note that, our scheme is not a simple application of the secure two-party computation techniques. In fact, these original protocols that we proposed are one of the notable contributions. Our contributions in this paper can be summarized as follows:

1) To our best knowledge, this is the first work that studies secure k-NN query on encrypted data with multiple keys. Our scheme not only preserves the data confidentiality and query privacy but also supports the offline data owner. Based on the property of multiple keys, we can thoroughly solve the problems induced by key-sharing with query users.
2) We present a set of novel protocols of secure two-party computation based on distributed two trapdoors public-key cryptosystem, which become a cornerstone of our secure k-NN scheme.
3) Based on the original protocols that we proposed, we construct a secure kNN scheme with multiple keys. And we show that the proposed scheme is secure under the standard semi-honest model [19]. Also, we demonstrate the practical applicability of our solution through extensive experiments using a real-world dataset.

The remainder of this paper proceeds as follows. Related works are surveyed in Section 2. Section 3 presents distributed two trapdoors public-key cryptosystem and k-dimensional tree as a background. We define our system architecture and design goals in Section 4. Section 5 presents the details of our scheme. The security analysis are carried out in Section 6. We evaluate experimentally the performance of the proposed scheme in Section 7. Finally, Section 8 concludes the paper and discusses future directions.

2 Related Work

Our work is a special type of query processing on encrypted data, which has gained much more focus recently, especially in the situation of cloud computing where the data owner outsources his data to the cloud. Existing works primarily concern the following aspects: traditional SQL query [20, 21], textual query [22, 23], range search [24, 25], top-k query [26–28] and k-NN query [8–10, 15–17, 29, 30]. In this section, we mainly review some recent achievements on secure k-NN query.

In previous works, some schemes have been proposed to solve secure k-NN computation on encrypted cloud data which can be mainly classified into two categories based on whether the key of data owner is sharing with others or not: key-sharing scheme and key-confidentiality scheme.

2.1 Secure k-NN Schemes with Key-sharing

In the key-sharing schemes, we assume that the query users are fully-trusted and know the key of the data owner. The data owner outsources his data and query functionality to the cloud where only trusted users are allowed to query the host data. Along this direction, researchers have proposed various methods to address secure k-NN problem.

Wong et al. [8] proposed a new encryption scheme (ASPE) that preserves the relative distances of all the database point to any query point that is sufficient to find k-NN. The ASPE transforms data points and queries with secret matrices, which are symmetric keys for the encryption scheme. Thus, it must be shared between the data owner and all query users. However, the method in [8] is not secure because it is prone to chosen-plaintext attacks [9].

To further improve the query performance, Yao et al. [9] designed a novel method based on secure Voronoi diagram algorithm. Instead of returning exact nearest neighbor, they allow a cloud server to return a relevant data partition. Furthermore, the work in [29] used Delaunay triangulation and order-preserving encryption [31] to solve the secure k-NN problem accurately. Although it can provide exact result, the solution incurs expensive overhead of computation and communication on the users. More importantly, the encryption schemes used in [9, 29] are symmetric, and the data owner and query users also have to share the key.

Unlike the model of the above schemes, Elmehdwi et al. [10] proposed a number of novel protocols over encrypted data based on a Twin-Cloud model [11] and Paillier cryptosystem [12], which can further increase security during query execution. They assume the existence of two semi-honest cloud servers P_1 and P_2 such that the encrypted data

TABLE 1
The Comparison of the Proposed Secure kNN Query Schemes

Schemes	Category	Resist CPA	Client Overhead
[8]	Key-sharing	No	low
[9]	Key-sharing	Yes	high
[29]	Key-sharing	Yes	low
[10]	Key-sharing	Yes	low
[30]	Key-sharing	Yes	high
[15]	key-confidentiality	No	low
[16]	key-confidentiality	No	low
[17]	key-confidentiality	Yes	low

is known only to P_1, whereas the secret key is just revealed to P_2. Using the secure protocols, P_1 collaborate with P_2 for the final result after receiving an encrypted query from the user. However this method also requires the users to access the private key of data owner and cannot be put into use for inefficiency.

Recently, Wang et al. [30] proposed a practical and secure nearest neighbor search on encrypted large-scale data. Specifically, the authors use order-preserving encryption [31, 32] to modify the search algorithm of nearest neighbors with tree structures for efficiency. Unfortunately, this scheme is designed to only achieve indistinguishability under ordered chosen-plaintext attacks (IND-OCPA) [32].

2.2 Secure k-NN schemes with key-confidentiality

In order to overcome the defects of key-sharing scheme, Zhu et al. [15] presented a new secure scheme with key-confidentiality. This work used a symmetric scheme with a secret matrix transformation as a key, and query users do not share this key. Instead, they interact with the data owner to derive a query encryption without revealing the query. This means that the data owner need to remain online for all the users. Recently, the work in [16] improves the scheme in [15] for supporting the offline data owner. However, the kernel of the above works [15, 16] is the ASPE mothod [8] while the ASPE method cannot be proved to resist the chosen-plaintext attack (CPA) [9]. The system in [17] is designed to solve the key-sharing problems by using a proxy server. Query users in this system encrypt the query point and send a request to proxy server instead of the cloud, then the proxy performs query operations. So this scheme can protect the confidentiality of the owners key to each user. However, it works based on the assumption that there is a fully-trusted proxy server between users and cloud server. Thus it cannot be practice in the real-world. The comparison of existing secure kNN query schemes are presented in Table 1.

3 PRELIMINARY

In this Section, we introduce essential preliminary concepts, such the Distributed Two Trapdoors Public-Key Cryptosystem (DT-PKC) [18] and k-d tree (k-dimensional tree) [33] in the literature, which will serve as the basic of our scheme. For ease of reference, Table 2 summarizes the key notations used throughout this paper.

3.1 Distributed Two Trapdoors Public-Key Cryptosystem (DT-PKC)

The Distributed Two Trapdoors Public-Key Cryptosystem (DT-PKC) [18] was adapted from a double trapdoor decryption cryptosystem [34]. The most prominent characteristic of DT-PKC is that each encrypted data in this can be decrypted by the strong trapdoor, and the strong private key is further protected by the secret sharing. The DT-PKC scheme works as follows:

KeyGen: On input a security parameter k and two large prime numbers p, q, where $\mathcal{L}(p) = \mathcal{L}(q) = k$, compute $N = pq$ and $\lambda = lcm(p-1, q-1)/2$. Define a function $L(x) = (x-1)/N$, choose a generator g of order $(p-1)(q-1)/2$, then randomly select $\theta_i \in [1, N/4]$ and compute $h_i = g^{\theta_i} \bmod N^2$ for party i. The public key $pk_i = (N, g, h_i)$, the corresponding weak private key $sk_i = \theta_i$, and strong private key $SK = \lambda$.

Encryption (Enc): The algorithm takes as input a public key pk_i and a message $m \in \mathbb{Z}_N$. It chooses a random $r \in [1, N/4]$ and outputs the ciphertext as $[m]_{pk_i} = \{T_{i,1}, T_{i,2}\}$, where $T_{i,1} = g^{r\theta_i}(1+mN) \bmod N^2$ and $T_{i,2} = g^r \bmod N^2$.

Decryption with weak private key (WDec): The algorithm $D_{sk_i}(\cdot)$ takes as input a ciphertext $[m]_{pk_i}$ and a weak private key $sk_i = \theta_i$ then outputs $m = L((T_{i,1} \cdot (T_{i,2}^{\theta_i})^{-1}) \bmod N^2)$.

Decryption with strong private key (SDec): Any ciphertext $[m]_{pk_i}$ can be decrypted using decryption algorithm $D_{SK}(\cdot)$ with strong private key $SK = \lambda$ by first calculating $T_{i,1}^{\lambda} \bmod N^2 = g^{\lambda\theta_i r} \cdot (1+mN\lambda) \bmod N^2 = 1+mN\lambda$. Then, m can be recovered as $m = L(T_{i,1}^{\lambda} \bmod N^2) \cdot \lambda^{-1} \bmod N$.

Strong private key splitting (SkeyS): The strong private key $SK = \lambda$ can be randomly split into two parts. The partial strong private keys $SK^{(i)} = \lambda_j (j = 1, 2)$, s.t., $\lambda_1 + \lambda_2 \equiv 0 \bmod \lambda$ and $\lambda_1 + \lambda_2 \equiv 1 \bmod N^2$ hold at the same time.

Partial decryption with partial strong private key step one (PSDec1): The algorithm $PDO_{SK^{(1)}}(\cdot)$ takes as input $[m]_{pk_i}$ and a partial strong private key $SK^{(1)} = \lambda_1$ then outputs the partial decrypted ciphertext $CT_i^{(1)} = T_{i,1}^{\lambda_1} = g^{r\theta_i\lambda_1} \cdot (1 + mN\lambda_1) \bmod N^2$.

Partial decryption with partial strong private key step two (PSDec2): The algorithm $PDT_{SK^{(2)}}(\cdot, \cdot)$ takes as input $CT_i^{(1)}$ and $[m]_{pk_i}$ then executes $CT_i^{(2)} = T_{i,1}^{\lambda_2} = g^{r\theta_i\lambda_2} \cdot (1 + mN\lambda_2) \bmod N^2$. At last, the algorithm computes $m = L(CT_i^{(1)} \cdot CT_i^{(2)})$.

Note that for given $m_1, m_2 \in \mathbb{Z}_N$ under the same pk, we have $[m_1]_{pk} \cdot [m_2]_{pk} = \{(1 + (m_1 + m_2) \cdot N) \cdot h^{r_1+r_2} \bmod N^2, g^{r_1+r_2} \bmod N^2\} = [m_1 + m_2]_{pk}$, and $[m]_{pk}^{N-1} = \{(1 + (N-1)m \cdot N) \cdot h^{(N-1)r_1} \bmod N^2, g^{(N-1)r_1} \bmod N^2\} = [-m]_{pk}$. In this paper, the numbers involved in our scheme are integer (i.e. m can be positive, negative or zero); therefore, we restrict m with the limit $\mathcal{L}(m) < \mathcal{L}(N)/8$.

3.2 k-d Tree

The k-d tree (k-dimensional tree) [33] is a space-partitioning data structure for organizing points in a k-dimensional space. The k-d tree is a useful data structure which can improve the search efficiency of range searches and kNN searches [35, 36]. The k-d tree is a binary tree in which every

TABLE 2
Notations and Definitions Used

Notations	Definitions
pk_o	Public key of data owner
pk_{u_i},sk_{u_i}	Public key of user i, Weak private key of user i
pk	Temporary public key for each query
SK	Strong private key
$SK^{(1)}$,$SK^{(2)}$	Partial strong private key
$[x]_{pk}$	Encrypted data x under pk
$D_{sk}(\cdot)$	Decryption algorithm using sk
$PDO_{SK^{(1)}}(\cdot)$	Partial decryption with $SK^{(1)}$
$PDT_{SK^{(2)}}(\cdot)$	Partial decryption with $SK^{(2)}$
$\mathcal{L}(x)$	Bit-length of x

node is a k-dimensional point. Every non-leaf node can be thought of as implicitly generating a splitting hyperplane that divides the space into two parts. Indeed, k-d trees use the "divide-and-conque" strategy to divide the space into several parts.

A simple example of the k-d tree is illustrated in Fig. 1. First, a vertical line (e.g. $x = 7$) splits the point set into two subsets of equal cardinality. Each of the resulting subsets is further split along a horizontal line (e.g. $y = 4$ and $y = 6$). The process will repeat until the cardinality of a node drops below a certain threshold. The two dimensional space in Fig. 1 was divided into seven parts in the end.

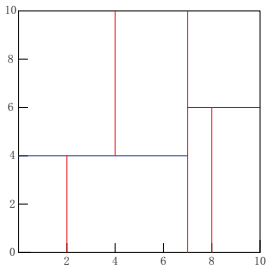


Fig. 1. A example of the k-d tree.

4 System Architecture and Design Goal

In this section, we briefly present the architecture of the secure kNN system and outline the threat model and design goal.

4.1 System Architecture

Our system architecture mainly consists of five kinds of entities: Key Generation Center (KGC), Cloud Platform (CP), Computation Service Provider (CSP), Data Owner (DO) and Query Users (QUs), shown in Fig. 2.

1) KGC: The trusted KGC is responsible for generating and managing both public and private keys in the system.
2) CP: A CP has abundant storage resources to store and manage data outsourced from all valid QUs. A CP also records all intermediate and final results in encrypted form in the process of protocol's implementation. In addition, a CP is able to perform certain computation over encrypted data.
3) CSP: A CSP provides online computation services in the system. So the CSP can offload the calculation task to CP and collaborates with it to find the k-NN for QUs in a privacy-preserving manner.
4) DO: Data are generated by the DO, encrypted using his public key and then outsourced to CP for storage.
5) QUs: The goal of a QU is to request the CP to perform secure k-NN query and get the encrypted result that can be decrypted by QU.

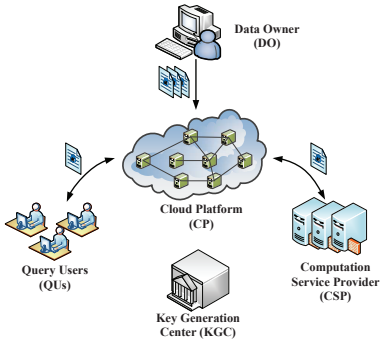


Fig. 2. System architecture.

Note that we assume that the authorization and the access control are well performed in our system. Actually, we can use an authentication scheme to verify the authenticity of the QUs, the details of the implementation can be referred to [37–41]. However, fully-trusted query users cannot be guaranteed through authorization mechanism. In other words, the users who have passed a verification step have the permissions to access the system. Yet, they are also very likely to attack the system. This is also the basic motivation of the paper. Our system introduces a CSP to produce a Twins-Clouds architecture compared with traditional single-cloud platform. The CSP is necessary in our system, on one hand a CP is not able to perform various compute operations efficiently (existing fully homomorphic cryptosystem [42] is rather inefficient, in term of computation and storage [43, 44]). On the other hand, Twins-Clouds architecture can minimize interactions between the users and cloud server while the only one cannot (due to the impossibility of program obfuscation [45]). In our scheme, users only need to send encrypted query initially and remain offline until retrieving encrypted outputs.

4.2 Threat Model

In the threat model, we assume the KGC to be a trusted entity, which generates the public and private keys for the system. On the other hand, CP, CSP, DO and QUs are semi-honest, (i.e., *honest-but-curious*). It means that these parties intend to follow the protocols strictly and return correct computation results, but try to infer the private information of other parties based on the data he receives and holds. Meanwhile, we also assume CP and CSP are non-colluding. To provide a specific security and privacy requirements, we

define the concrete data confidentiality and query privacy to against active adversary $\mathcal{A}$ as follows.

Definition 1 (Data Confidentiality Definition). *$\mathcal{A}$ cannot learn any plaintext of the encrypted database stored in the CP.*

Definition 2 (Query Privacy Definition). *Neither the query point nor the result (k-NN) for users should be reveal to the $\mathcal{A}$.*

To satisfy these requirements, the active adversary $\mathcal{A}$ in the threat model has the following attacking abilities:

1) $\mathcal{A}$ may eavesdrop all the communication links to get the encrypted data.
2) $\mathcal{A}$ may compromise the CP and try to crack the encrypted database outsourced by the DO.
3) $\mathcal{A}$ may compromise the CSP to guess the plaintext of the all encrypted data sent from CP by executing interactive protocols.
4) $\mathcal{A}$ may compromise one or more QUs to obtain their decryption abilities.

The adversary $\mathcal{A}$ is, however, restricted from compromising the CP and the CSP at the same time. We remark that such restriction is typical in adversary models used in cryptographic protocols [10, 11, 18]. In addition, the access pattern, which is the sequence of results, is not considered to be protected in our scheme due to the extremely high complexity, i.e. to protect it, the algorithm has to touch the whole dataset [46].

Existing works [8–10, 29, 30] cannot resist the above adversary $\mathcal{A}$, since the data owner in these schemes will share the decryption key with query users; once $\mathcal{A}$ compromises any query user, the outsourced data can be leaked entirely. However, in our scheme, if a query user was captured by $\mathcal{A}$, only the compromised user's query privacy would be disclosed, while the privacy of other users and the confidentiality of the cloud data would be preserved.

4.3 Design Goal

In this paper, our method will fulfill privacy and performance guarantees as follows:

1) Data confidentiality and query privacy: the data confidentiality and query privacy as described in the Definition 1 and 2 should be guaranteed.
2) Low computation overhand on the QUs: the QUs in our system generally have limited computation and communication resources, our method should be designed for reducing the users' overhand.
3) Support offline data owner: a large number of QUs are involved in the system, therefore supporting offline data owner is quite necessary in terms of the system's scalability.

5 Secure k-NN Query with Multiple Keys

5.1 Overview

In this paper, we introduce a secure k-NN query scheme with multiple keys. Let D denote data owner's original database. We assume DO's database consists of n records, denoted by $D = \{p_1, p_2, \ldots, p_n\}$, and each point is a m-dimensional vector, i.e. $p_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,m})$, for all $i = 1, 2, \ldots, n$. Initially, DO encrypts his database locally, that is, he computes $[p_{i,j}]_{pk_o}$, for $1 \le i \le n$ and $1 \le j \le m$. Let the encrypted database be denoted by D'. We assume that DO outsources D' as well as the future query processing service to the CP. Each QU holds some private m-dimensional query points. For the query point $q = (q_1, q_2, \ldots, q_m)$, QU would like to retrieve the top k records that are closest to the query point in D according to the Euclidean distance, that is, $\|p_i, q\| = \sqrt{\sum_{j=1}^{m} (p_{i,j} - q_j)^2}$. QU initially sends his query q (in encrypted form) to CP. After this, CP and CSP involve in a set of sub-protocols to securely compute the Euclidean distance then retrieve the k-NN in D' and return encrypted result to the QU. At the end of our scheme, only the corresponding query user can decrypt the result points.

5.2 Scheme Details

In this section, we will describe the construction of our scheme step-by-step.

5.2.1 Setup

Recall that Section 2 describes a novel cryptosystem DT-PKC. In our system, we have one DO and η QU. The KGC generates a public key $pk_o = (N, g, h_o = g^{\theta_o})$ and a public-private key pair $pk_{u_i} = (N, g, h_i = g^{\theta_i})$ and $sk_{u_i} = \theta_i (i = 1, \ldots, \eta)$ under the same N and g for the DT-PKC; the KGC sends pk_o to DO and CP, then distributes the individual public-private key pair pk_{u_i}/sk_{u_i} to each QU. In addition, the KGC generates a temporary public key $pk = (N, g, h = g^{\theta})$ for each query then send pk and $pk_{u_i} (i = 1, \ldots, \eta)$ to CP and CSP. Moreover, the strong private key SK is randomly spit into $SK^{(1)}$ and $SK^{(2)}$ using **SkeyS** algorithm, sending to CP and CSP for storage respectively. For readability, if the public/private key are associated with the QU i, we will omit the secondary subscript i from the symbols (i.e., use pk_u/sk_u instead of pk_{u_i}/sk_{u_i}).

After receiving pk_o from KGC, DO uses it to encrypt each record p_i in database D, i.e. $[p_i]_{pk_o} = ([p_{i,1}]_{pk_o}, [p_{i,2}]_{pk_o}, \ldots, [p_{i,m}]_{pk_o})$, for all $i = 1, 2, \ldots, n$, and outsources the encrypted database D' to CP. Next, the query user encrypts the query point q with his own public key, this is, computes $[q]_{pk_u} = ([q_1]_{pk_u}, [q_2]_{pk_u}, \ldots, [q_m]_{pk_u})$ and send it to CP. So far, the work of DO is completed and DO can remain offline.

5.2.2 Secure Euclidean Distance Computation (SecDist)

Since the basic of our scheme is a function of distance, we must be able to compute distance using encrypted data without revealing it. Algorithm 1 describes secure Euclidean distance computation — SecDist$([p_i]_{pk_o}, [q]_{pk_u})$, which allows the CP to get $[\|p_i - q\|^2]_{pk}$, where $\|p_i - q\|$ denotes the Euclidean distance between p_i and q. The protocol does not leak any information about p_i and q to the CP and CSP.

The basic idea of **SecDist** is based on the following equation:

$$(p_{i,j} - q_j)^2 = (p_{i,j} - q_j + R)^2 - 2R \cdot (p_{i,j} - q_j) - R^2 \quad (1)$$

The overall steps in **SecDist** are shown in Algorithm 1. In Step 1, CP initially randomizes $p_{i,1}$ and q_1 by computing ciphertexts $X = [p_{i,1} + r_1]_{pk_o}$ and $Y = [q_1 + r_2]_{pk_u}$, and uses partial strong private key $SK^{(1)}$ to partially decrypt them,

then send X, Y and the partially ciphertexts X_1 and Y_1 to CSP. Upon receiving, CSP decrypts them with $SK^{(2)}$ to get $X_2 = p_{i,1} + r_1, Y_2 = q_1 + r_2$. Then, CSP encrypts $(X_2 - Y_2)$ and $(X_2 - Y_2)^2$ with the temporary public key pk and sends them to CP. After this, CP removes extra random factors from $(p_{i,j} - q_j + R)^2$ based on Equation (1) to get $Z_1 = [(p_{i,1} - q_1)^2]_{pk}$. To compute $Z_j (j = 2, \ldots, m)$, repeat Step 1 to Step 3 that we followed to compute Z_1. Finally, the CP can compute $[\|p_i - q\|^2]_{pk}$ as in Step 5. Note that the ciphertexts of the distances are transformed into ones encrypted under the same key pk after the process of **SecDist**.

5.2.3 Secure k-NN Retrieve

Now that we can compute Euclidean distances, we must compute the top-k (in encrypted form) records that are closest to q in a secure manner. For that, we firstly present two secure protocols — Secure Minimum (**SecMin**) and Secure Minimum Index of n numbers (**SecMI**$_n$), based here to build the secure k-NN retrieve (**Sec**k**NN**) scheme.

Secure Minimum (SecMin). The goal of SecMin$(([a_i]_{pk}, [i]_{pk}), ([a_j]_{pk}, [j]_{pk}))$ is for CP and CSP jointly compute the encryption and the encrypted index of the minimum number between a_i and a_j, which will be known only to CP. For example, when $a_1 = 7, a_2 = 3$, $SecMin(([a_1]_{pk}, [1]_{pk}), ([a_2]_{pk}, [2]_{pk}))$ will only return $[3]_{pk}$ and $[2]_{pk}$ to CP. The **SecMin** protocol can be departed into two part. The first (refer to Step 1 to Step 3(a) in Algorithm 2) is to obtain the encrypted data $[u^*]_{pk}$ to show the relationship between the plaintext of the two encrypted data (i.e. $a_i \geq a_j$ or $a_i \leq a_j$). More specifically, CP flips a coin s randomly, if $s = 1$, CP calculates $[l]_{pk} = [a_i - a_j]_{pk}$ otherwise $[l]_{pk} = [a_j - a_i]_{pk}$, then masks the value $[l]_{pk}$ with a positive random r. In Step 2, CSP decides the value of u (0 or 1) according to the $r \cdot l$ (larger or smaller than zero) and return the $[u]_{pk}$ to CP. Here, the condition $\mathcal{L}(X_2) > \mathcal{L}(N)/2$ is satisfied only when $r \cdot l < 0$, because of the limitation on value ranges in DT-PKC (see [18] for more detailed explanation). Once $[u]_{pk}$ is received, CP computes as follows: if $s = 1$, CP denotes $[u^*]_{pk} = [u]_{pk}$. Otherwise, CP computes $[u^*]_{pk} = [1]_{pk} \cdot ([u]_{pk})^{N-1} = [1 - u]_{pk}$. By this stage if $u^* = 0$, it shows $[a_i]_{pk} \geq [a_j]_{pk}$; and if $u^* = 1$, it shows $[a_i]_{pk} < [a_j]_{pk}$.

The second part of the **SecMin** protocol (refer to Step 3(b) to Step 6 in Algorithm 2) is to compute the encryption of the minimum value between a_i and a_j, i.e., $[min(a_i, a_j)]_{pk}$, using the following formulation equation:

$$min(a_i, a_j) = a_j + u^* \cdot (a_i - a_j) \tag{2}$$

CP and CSP jointly compute the encryption of $(u^* \cdot (a_i - a_j))$, the basic idea is as following:

$$u^* \cdot (a_i - a_j) = (u^* + r_1) \cdot (a_i - a_j + r_2) - r_1 \cdot (a_i - a_j) - r_2 \cdot u^* - r_1 \cdot r_2 \tag{3}$$

The detailed computation process is analogous to one of the Equation (1) (and not explained here). After that, CP can locally calculates $[min(a_i, a_j)]_{pk} = [a_j + u^*(a_i - a_j)]_{pk}$. In a similar manner, apart from the encrypted minimum value, CP and CSP can compute $[l_{min(a_i, a_j)}]_{pk}$ (i.e. the encrypted index associated with the minimum value) in Step 6.

Secure Minimum Index of n numbers (SecMI$_n$). Suppose $[a_1]_{pk}, \ldots, [a_n]_{pk}$ denote the list of n encrypted integers and i denotes the index of integer a_i in the list, for $1 \leq i \leq n$. The **SecMI**$_n$ protocol computes the index of minimum value $[l_{min(a_i, \ldots, a_n)}]_{pk}$ without revealing the plaintext of a_i to CP and CSP. Here we construct the **SecMI**$_n$ protocol using the **SecMin** protocol as a building block. Step 1(a) computes $T = [min(a_1, a_2)]_{pk}$ and $I = [l_{min(a_1, a_2)}]_{pk}$ (known only to CP) using the **SecMin** protocol. After this, CP with input (T, I) and $([a_3]_{pk}, [3]_{pk})$ can calculate $[min(a_1, a_2, a_3)]_{pk}$ and $[l_{min(a_1, a_2, a_3)}]_{pk}$ in the same way. The above process is repeated until CP gets $(T, I) = ([min(a_1, \ldots, a_n)]_{pk}, [l_{min(a_1, \ldots, a_n)}]_{pk})$. Then, in Step 2 - Step 4, CP and CSP jointly decrypt the encrypted index I to get the plaintext index Γ. A complete process is shown in Algorithm 3.

Based on the above algorithms, we describe the details of our secure k-NN retrieve (Sec$k$NN) scheme in Algorithm 4. This algorithm takes as input a original database D and a query point q then outputs the result points T. In Step 1, the CP and CSP jointly involve in **SecDist** protocol to compute $[d_i]_{pk} (1 \leq i \leq n)$, then CP uses $\overrightarrow{d}$ to denote $\{([d_1]_{pk}, 1), \ldots, ([d_n]_{pk}, n)\}$ in Step 2. As was previously mentioned, $\overrightarrow{d}$ is revealed only to CP. After this, CP and CSP obtain the index Γ_1 corresponding to the minimum by using **SecMI**$_n$ protocol, then remove $([d_{\Gamma_1}]_{pk}, \Gamma_1)$ from $\overrightarrow{d}$. The above process described in Step 3 is repeated until k iterations, and in each iteration Γ_j corresponds to the index of the minimum value in the current $\overrightarrow{d}$. As indicated in Step 4(a), CP will get a index list $\Gamma = (\Gamma_1, \Gamma_2, \ldots, \Gamma_k)$ at the end of the iteration.

Next, CP proceeds as follows: select $[p_{\Gamma_1}]_{pk_o}, [p_{\Gamma_2}]_{pk_o}, \ldots, [p_{\Gamma_k}]_{pk_o}$ as the k-NN result based on Γ, and mask each with a random number, such that $t_{j,h} = [p_{\Gamma_j, h}]_{pk_o} \cdot [r_{j,h}]_{pk_o}$, for $1 \leq j \leq k$ and $1 \leq h \leq m$, where $p_{\Gamma_j, h}$ denotes the h^{th} dimensional value of record p_{Γ_j}, m and k represent the dimension and the number of result points, respectively. This prevents to leak $p_{\Gamma_j, h}$ to

---

Algorithm 1 SecDist$([p_i]_{pk_o}, [q]_{pk_u}) \rightarrow [\|p_i - q\|^2]_{pk}$

Require: CP has $([p_i]_{pk_o}, [q]_{pk_u}), SK^{(1)}, pk, pk_u, pk_o$; CSP has $SK^{(2)}, pk$.

1. CP:
   (a) Choose random $r_1, r_2 \in \mathbb{Z}_N (r_1 \neq r_2)$, compute $R = r_1 - r_2$.
   (b) $X \leftarrow [p_{i,1}]_{pk_o} \cdot [r_1]_{pk_o}, Y \leftarrow [q_1]_{pk_u} \cdot [r_2]_{pk_u}$, $X_1 \leftarrow \mathbf{PDO}_{SK^{(1)}}(X), Y_1 \leftarrow \mathbf{PDO}_{SK^{(1)}}(Y)$.
   (c) Send X, X_1, Y, Y_1 to CSP.
2. CSP:
   (a) $X_2 \leftarrow \mathbf{PDT}_{SK^{(2)}}(X_1, X), Y_2 \leftarrow \mathbf{PDT}_{SK^{(2)}}(Y_1, Y)$.
   (b) $S_1 \leftarrow [X_2 - Y_2]_{pk}, S_2 \leftarrow [(X_2 - Y_2)^2]_{pk}$.
   (c) Send S_1, S_2 to CP.
3. CP:
   (a) $S_3 \leftarrow S_1 \cdot [R]_{pk}^{N-1}$.
   (b) $Z_1 \leftarrow S_2 \cdot S_3^{N-2R} \cdot [R^2]_{pk}^{N-1} = [(p_{i,1} - q_1)^2]_{pk}$.
4. CP and CSP:
   (a) Repeat Step 1 – Step 3 to calculate $Z_2 = [(p_{i,2} - q_2)^2]_{pk}, \ldots, Z_m$.
5. CP:
   (a) $[\|p_i - q\|^2]_{pk} \leftarrow \prod_{j=1}^{m} Z_j$.

Algorithm 2 SecMin$(([a_i]_{pk}, [i]_{pk}), ([a_j]_{pk}, [j]_{pk})) \rightarrow ([min(a_i, a_j)]_{pk}, [l_{min(a_i,a_j)}]_{pk})$

Require: CP has $([a_i]_{pk}, [i]_{pk}), ([a_j]_{pk}, [j]_{pk}), SK^{(1)}, pk$; CSP has $SK^{(2)}, pk$.

1. CP:
   (a) Flip a coin s randomly(i.e. $s = 1$ or $s = 0$).
   (b) **if** $s = 1$ **then**
   $[l]_{pk} = [a_i]_{pk} \cdot [a_j]_{pk}^{N-1}$
   else
   $[l]_{pk} = [a_j]_{pk} \cdot [a_i]_{pk}^{N-1}$
   end if
   (c) Chooses a random number r, s.t. $\mathcal{L}(r) < \mathcal{L}(N)/4$, $X = [l]_{pk}^{r} = [r \cdot l]_{pk}, X_1 = \mathbf{PDO}_{SK^{(1)}}(X)$.
   (d) Send X, X_1 to CSP.
2. CSP:
   (a) $X_2 \leftarrow \mathbf{PDT}_{SK^{(2)}}(X_1, X)$.
   (b) **if** $\mathcal{L}(X_2) > \mathcal{L}(N)/2$ **then**
   $u = 1$
   else
   $u = 0$
   end if
   (c) Encrypt u with pk and send $[u]_{pk}$ to CP.
3. CP:
   (a) **if** $s = 1$ **then**
   $[u^*]_{pk} = [u]_{pk}$
   else
   $[u^*]_{pk} = [1]_{pk} \cdot [u]_{pk}^{N-1} = [1 - u]_{pk}$
   end if
   (b) Choose random $r_1, r_2 \in \mathbb{Z}_N$.
   (c) $Y = [u^*]_{pk} \cdot [r_1]_{pk}, T = [a_i]_{pk} \cdot [a_j]_{N-1}^{pk} \cdot [r_2]_{pk}$, $Y_1 \leftarrow \mathbf{PDO}_{SK^{(1)}}(Y), T_1 \leftarrow \mathbf{PDO}_{SK^{(1)}}(T)$.
   (d) Send Y, Y_1, T, T_1 to CSP.
4. CSP:
   (a) $Y_2 \leftarrow \mathbf{PDT}_{SK^{(2)}}(Y_1, Y), T_2 \leftarrow \mathbf{PDT}_{SK^{(2)}}(T_1, T)$.
   (b) $S \leftarrow Y_2 \cdot T_2$.
   (c) Send $[S]_{pk}$ to CP.
5. CP:
   (a) $S_1 \leftarrow [S]_{pk} \cdot ([a_i]_{pk} \cdot [a_j]_{pk}^{N-1})^{N-r_1} \cdot [u^*]_{pk}^{N-r_2} \cdot [r_1 - r_2]_{pk}^{N-1} = [u^* \cdot (a_i - a_j)]_{pk}$.
   (b) $[min(a_i, a_j)]_{pk} \leftarrow [a_j]_{pk} \cdot S_1 = [a_j + u^* \cdot (a_i - a_j)]_{pk}$.
6. CP and CSP:
   (a) Calculate $[l_{min(a_i,a_j)}]_{pk} = [j + u^* \cdot (i - j)]_{pk}$ refer to Step 3(b) – Step 5.

the CSP. Use $SK^{(1)}$ to partially decrypt $t_{j,h}$, then send $t_{j,h}$ and the partially ciphertext $t'_{j,h}$ to CSP in Step 4(b). Upon receiving $t_{j,h}$ and $t'_{j,h}$, for $1 \le j \le k$ and $1 \le h \le m$, CSP decrypts them to get $t''_{j,h} = p_{\Gamma_j,h} + r_{j,h}$. Then CSP encrypts $t''_{j,h}$ with QU's public key pk_u to get $[t''_{j,h}]_{pk_u}$ and sends them to CP, as in Step 5. Step 6 shows that CP removes the random factors from $[t''_{j,h}]_{pk_u} (1 \le j \le k, 1 \le h \le m)$ by computing $T_{j,h} = [t''_{j,h}]_{pk_u} \cdot [r_{j,h}]_{pk_u}^{N-1} = [p_{\Gamma_j,h}]_{pk_u}$, and Send $T_{j,h}$ to QU. Finally, QU decrypts them with sk_u by **WDec** algorithm (described in Section 3.1) to get $p_{\Gamma_j,h}$, for $1 \le j \le k$ and $1 \le h \le m$. At this point, $p_{\Gamma_j,h} (1 \le j \le k)$ are the j^{th} nearest neighbor to the query q.

We remark that, to facilitate the understanding, our Sec$k$NN scheme actually adopted a relatively direct way of comparing one by one (refer to Step 3) when retrieving

Algorithm 3 SecMI$_n(([a_1]_{pk}, [1]_{pk}), \ldots, ([a_n]_{pk}, [n]_{pk})) \rightarrow \Gamma$

Require: CP has $([a_1]_{pk}, [1]_{pk}), \ldots, ([a_n]_{pk}, [n]_{pk}), SK^{(1)}, pk$; CSP has $SK^{(2)}, pk$.

1. CP and CSP:
   (a) $(T, I) \leftarrow$ **SecMin**$(([a_1]_{pk}, [1]_{pk}), ([a_2]_{pk}, [2]_{pk}))$.
   (b) **for** $i = 2$ to $n - 1$ **do**
   $(T, I) \leftarrow$ **SecMin**$((T, I), ([a_{i+1}]_{pk}, [i + 1]_{pk}))$.
   end for
2. CP:
   (a) $I_1 = \mathbf{PDO}_{SK^{(1)}}(I)$.
   (b) Send I, I_1 to CSP.
3. CSP:
   (a) $I_2 \leftarrow \mathbf{PDT}_{SK^{(2)}}(I_1, I)$.
   (b) Send I_2 to CP.
4. CP:
   (a) $\Gamma \leftarrow I_2$.

k-NN. The computational complexity of retrieval algorithm is $O(n \cdot k)$. However, all ranking-based methods such as merging sort, quicksort and heapsort can be used in our scheme to enhance the efficiency, e.g. based quicksort retrieving complexity can reach $O(k \cdot \log n)$.

5.3 Optimization

Our scheme runs on the top of encrypted data for the secure kNN query, whereas it does introduce inefficiency. Now we discuss two strategies to boost the efficiency. One is the k-d tree [33] and the other is offline and pipeline computing [47].

5.3.1 k-d Tree

Since k-d trees divide the range of a domain in half at each level of the tree, they are useful for performing an optimization of the spatial query. For ease of description, we assume each data record (a point) has two dimensions (i.e. m = 2). Next we will explain the optimization in the context of the point set partition shown in Fig. 3.

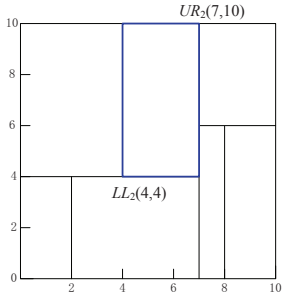


Fig. 3. Partitioning data points using a k-d tree.

The DO preprocesses the source data according to a k-d tree decomposition, each rectangular partition in Fig. 3 has roughly $n/7$ data points, and can be uniquely identified by its lower-left (LL) and upper-right (UR) corners. For instance, the 2^{nd} partition is enclosed by $LL_2(4, 4), UR_2(7, 10)$. DO encrypts each LL/UR corners and sent to CP. When receiving a encrypted query $[q]_{pk_u}$

Algorithm 4 Sec$k$NN$(D, [q]_{pk}) \rightarrow T$

Require: CP holds $D, [q]_{pk}, SK^{(1)}, pk, pk_u, pk_o$;
CSP has $SK^{(2)}, pk, pk_u$; QU has sk_u.

1. CP and CSP:
   (a) **for** $i = 1$ to n **do**
   $[d_i]_{pk} \leftarrow$ **SecDist**$([p_i]_{pk_o}, [q]_{pk_u}) = [\|p_i - q\|^2]_{pk}$.
   end for
2. CP:
   (a) Use $\overrightarrow{d}$ to denote $\{([d_1]_{pk}, 1), \ldots, ([d_n]_{pk}, n)\}$.
3. **for** $j = 1$ to k **do**
   (a) CP and CSP:
   $\Gamma_j \leftarrow$ **SecMI**$_{n-j+1}(\overrightarrow{d})$.
   (b) CP:
   Remove $([d_{\Gamma_j}], \Gamma_j)$ from $\overrightarrow{d}$.
   end for
4. CP:
   (a) Get $\Gamma = (\Gamma_1, \Gamma_2, \ldots, \Gamma_k)$.
   (b) **for** $1 \le j \le k$ and $1 \le h \le m$ **do**
   i) $t_{j,h} \leftarrow [p_{\Gamma_j,h}]_{pk_o} \cdot [r_{j,h}]_{pk_o}$, where $r_{j,h} \in \mathbb{Z}_N$.
   ii) $t'_{j,h} \leftarrow$ **PDO**$_{SK^{(1)}}(t_{j,h})$.
   iii) Send $t_{j,h}, t'_{j,h}$ to CSP.
   end for
5. CSP:
   (a) **for** $1 \le j \le k$ and $1 \le h \le m$ **do**
   i) $t''_{j,h} \leftarrow$ **PDT**$_{SK^{(2)}}(t'_{j,h}, t_{j,h}) = p_{\Gamma_j,h} + r_{j,h}$.
   ii) Send $[t''_{j,h}]_{pk_u}$ to CP.
   end for
6. CP:
   (a) **for** $1 \le j \le k$ and $1 \le h \le m$ **do**
   i) $T_{j,h} \leftarrow [t''_{j,h}]_{pk_u} \cdot [r_{j,h}]^{N-1}_{pk_u} = [p_{\Gamma_j,h}]_{pk_u}$.
   ii) Send $T_{j,h}$ to QU.
   end for
7. QU:
   (a) **for** $1 \le j \le k$ and $1 \le h \le m$ **do**
   $p_{\Gamma_j,h} \leftarrow \mathbf{D}_{sk_u}(T_{j,h})$.
   end for
   (b) $T = \{p_{\Gamma_j,h} / 1 \le j \le k, 1 \le h \le m\}$.

Algorithm 5 SecComp$([a]_{pk_o}, [b]_{pk_u}) \rightarrow u^*(a < b$ **iff** $u^* = 1)$

Require: CP has $[a]_{pk_o}, [b]_{pk_u}, SK^{(1)}, pk, pk_u, pk_o$;
CSP has $SK^{(2)}, pk$.

1. CP:
   (a) Choose random $r_1, r_2 \in \mathbb{Z}_N (r_1 \neq r_2)$, compute $R = r_1 - r_2$.
   (b) $X \leftarrow [a]_{pk_o} \cdot [r_1]_{pk_o}, Y \leftarrow [b]_{pk_u} \cdot [r_2]_{pk_u}$,
   $X_1 \leftarrow$ **PDO**$_{SK^{(1)}}(X), Y_1 \leftarrow$ **PDO**$_{SK^{(1)}}(Y)$.
   (c) Send X, X_1, Y, Y_1 to CSP.
2. CSP:
   (a) $X_2 \leftarrow$ **PDT**$_{SK^{(2)}}(X_1, X)$.$Y_2 \leftarrow$ **PDT**$_{SK^{(2)}}(Y_1, X)$.
   (b) $S_1 \leftarrow [X_2 - Y_2]_{pk}, S_2 \leftarrow [Y_2 - X_2]_{pk}$.
   (c) Send S_1, S_2 to CP.
3. CP:
   (a) Flip a coin s randomly(i.e. $s = 1$ or $s = 0$).
   (b) **if** $s = 1$ **then**
   $[l]_{pk} = S_1 \cdot [R]^{N-1}_{pk}$
   else
   $[l]_{pk} = S_2 \cdot [R]_{pk}$.
   end if
   (c) Chooses a random number r, s.t. $\mathcal{L}(r) < \mathcal{L}(N)/4$,
   $Z = [l]^r_{pk}, Z_1 =$ **PDO**$_{SK^{(1)}}(Z)$.
   (d) Send Z, Z_1 to CSP.
4. CSP:
   (a) $Z_2 \leftarrow$ **PDT**$_{SK^{(2)}}(Z_1, Z)$.
   (b) **if** $\mathcal{L}(Z_2) > \mathcal{L}(N)/2$ **then**
   $u = 1$
   else
   $u = 0$.
   end if
   (c) Send u to CP.
5. CP:
   (a) **if** $s = 1$ **then**
   $u^* = u$
   else
   $u^* = 1 - u$.
   end if

from the QU, the CP finds a subspace which contains the query point by following test:

$$\begin{aligned} x_{LL} < x_q < x_{UR}, \\ y_{LL} < y_q < y_{UR}. \end{aligned} \tag{4}$$

Only satisfied with these two conditions at the same time, the partition contains the query point. Since LL/UR corners and query point are encrypted by pk_o and pk_u respectively, we provide the **SecComp** protocol to compare the ciphertexts in a multiple keys manner. A detailed description is presented in Algorithm 5. The goal of SecComp$([a]_{pk_o}, [b]_{pk_u})$ is for CP and CSP jointly compute the flag u^* to indicate the smaller of a and b (i.e. $a < b$**iff**$u^* = 1$), which will be known only to CP. The idea of this protocol is similar to **SecMin** algorithm, hence, it won't be covered again here.

Now that we can find out the eligible partition, then the CP just needs to execute the secure k-NN query in it. We remark that, in general, more than one partition might be accessed for the absolutely accurate results. Even so, this greatly reduces overhead and helps improve overall performance.

5.3.2 Offline and Pipeline Computing

It can be seen from the above that the k-d tree can trim the database size. Next we discuss how to use offline and pipeline computing to improve the processing power in our scheme.

In the DT-PKC cryptosystem [18], encryption of a integer $x \in \mathbb{Z}_N$ is given by $[x]_{pk} = \{g^{r\theta_i}(1 + xN) \bmod N^2; g^r \bmod N^2\}$, where r is a random number. Since the computation of $g^{r\theta_i} \bmod N^2$ and $g^r \bmod N^2$ is independent of x, it could be put into an offline phase. Furthermore, the actual online computation costs (with an offline phase) of our protocols can be much less than their costs without an offline phase. For example, Step 2 in **SecDist** protocol computes $X = [p_{i,1}]_{pk_o} \cdot [r_1]_{pk_o}$ and $Y = [q_1]_{pk_u} \cdot [r_2]_{pk_u}$, where r_1 and r_2 are randomly chosen by CP. It is obvious that the computation of $[r_1]_{pk_o}$ and $[r_2]_{pk_u}$ is independent of any specific factor of **SecDist**. That is, CP can precompute these values during the offline phase, thus reducing its online computation time. In a similar manner, CP and CSP can precompute certain intermediate values in each protocols.

We are able to further reduce the online execution time

by adopting the technique of pipeline computing. Take the execution of **SecDist** for instance, by which CP would like to compute $[(p_{i,1}-q_1)^2]_{pk}$ and $[(p_{i,2}-q_2)^2]_{pk}$. Here CP does not have to wait the response of CSP after sending $[p_{i,1}+r_1]_{pk_o}$ and $[q_1+r_2]_{pk_u}$. Instead, after sending $[p_{i,1}+r_1]_{pk_o}$ and $[q_1+r_2]_{pk_u}$ to CSP, CP can immediately start computing $[p_{i,2}+r_1]_{pk_o}$ and $[q_2+r_2]_{pk_u}$. We expect that we could further save at least one-third of the online execution time in the long run when the computation of $[(p_{i,j}-q_j)^2]_{pk}(1 \leq j \leq m)$ are executed in the pipeline. Likewise, we could pipeline the $\mathbf{SecMI}_n$ protocol to save much time.

6 Security Analysis

In this section, we analyze the security of the sub-protocols including in **SecDict**, **SecMin**, $\mathbf{SecMI}_n$, **SecComp** and then demonstrate our **Sec$k$NN** scheme can preserve the data confidentiality and query privacy against $\mathcal{A}$ described in Section 4.2.

6.1 Security of Sub-protocols

In order to prove the security of the sub-protocols, we first list the security theorem of the DT-PKC and the security definition of the protocol under semi-honest model as follows.

Theorem 1 [18]. *The DT-PKC scheme introduced in Section 3.1 is semantically secure, based on the assumed intractability of the DDH assumption over $\mathbb{Z}_{N^2}$.* (The details of the proof can be referred to [18])

Definition 3 (security in the semi-honest model) [19]. *Let a_i be the input of party P_i, $\Pi_i(\pi)$ be P_i's execution image of the protocol π and b_i be the output for party P_i computed from π. Then, π is secure if $\Pi_i(\pi)$ can be simulated from a_i and b_i such that distribution of the simulated image is computationally indistinguishable from $\Pi_i(\pi)$.* (Detailed security definition can be found in [19])

As mentioned above, we need to show that the simulated execution image of these sub-protocols are computationally indistinguishable from their actual execution image. Note that, the execution image generally includes the messages exchanged and the information computed from these messages.

Theorem 2. *The **SecDist** protocol described in Algorithm 1 is secure under semi-honest model.*

Proof. Here, let the execution image of CSP be denoted by $\Pi_{CSP}(SecDist)$ which is given by $\Pi_{CSP}(SecDist) = \{(X, X_1, X_2), (Y, Y_1, Y_2)\}$ where $X_2 = p_{i,j} + r_1$ and $Y_2 = q_j + r_2$ are derived by decrypting (X, X_1) and (Y, Y_1), respectively. Note that r_1 and r_2 are random numbers in $\mathbb{Z}_N$. We assume $\Pi^S_{CSP}(SecDist)$ means the simulated image of CSP, and $\Pi^S_{CSP}(SecDist) = \{(X', X'_1, X'_2), (Y', Y'_1, Y'_2)\}$ where all the elements are randomly generated from $\mathbb{Z}_N$. Since DT-PKC is a semantically secure encryption scheme, (X, X_1) and (Y, Y_1) are computationally indistinguishable from (X', X'_1) and (Y', Y'_1), respectively. Meanwhile, as X'_2 and Y'_2 are randomly chosen from $\mathbb{Z}_N$, X_2 and Y_2 are computationally indistinguishable from X'_2 and Y'_2, respectively. Based on the above, we can draw a conclusion that $\Pi_{CSP}(SecDist)$ is computationally indistinguishable from $\Pi^S_{CSP}(SecDist)$.

Similarly, the execution image of CP in **SecDist** protocol is given by $\Pi_{CP}(SecDist) = \{S_1, S_2\}$ where S_1 and S_2 are encrypted values. Let the simulated image of CP be given by $\Pi^S_{CP}(SecDist) = \{S'_1, S'_2\}$, where S'_1 and S'_2 are randomly chosen from $\mathbb{Z}_N$. Due to the semantically secure of DT-PKC, S_1 and S_2 are computationally indistinguishable from S'_1 and S'_2, respectively. As a result, $\Pi_{CP}(SecDist)$ is computationally indistinguishable from $\Pi^S_{CP}(SecDist)$. Combining the above analyses and associating with the Definition 3, we can confirm that **SecDist** protocol is sure under the semi-honest model.

Theorem 3. *The **SecMin** protocol described in Algorithm 2 is secure under semi-honest model.*

Proof. According to Algorithm 2, let the execution image of **SecMin** for CSP be denoted by $\Pi_{CSP}(SecMin)$ which is given by $\Pi_{CSP}(SecMin) = \{(X, X_1, X_2), (Y, Y_1, Y_2), (T, T_1, T_2), u\}$ where $X_2 = r \cdot l$, $Y_2 = u^* + r_1$ and $T_2 = a_i - a_j + r_2$ are separately derived by decrypting (X, X_1), (Y, Y_1) and (T, T_1). Note that r is random numbers in $\mathbb{Z}_{N^+}$ while r_1 and r_2 are randomly chosen from $\mathbb{Z}_N$. In addition, u denotes the comparison result compute from X_2. We assume $\Pi^S_{CSP}(SecMin)$ means the simulated image of CSP, and $\Pi^S_{CSP}(SecMin) = \{(X', X'_1, X'_2), (Y', Y'_1, Y'_2), (T', T'_1, T'_2), u'\}$, where (X', X'_1, X'_2), (Y', Y'_1, Y'_2) and (T', T'_1, T'_2) are randomly generated from $\mathbb{Z}_N$ whereas u' is set to 0 or 1 according to the randomly tossed coin. Since DT-PKC is a semantically secure encryption scheme, (X, X_1, X_2), (Y, Y_1, Y_2) and (T, T_1, T_2) are computationally indistinguishable from (X', X'_1, X'_2), (Y', Y'_1, Y'_2) and (T', T'_1, T'_2), respectively. Furthermore, because the element s is a coin flipped randomly by CP (at Step 1(a) in Algorithm 2), u is either 0 or 1 with equal probability. Thus, u is computationally indistinguishable from u'. Combining the above results, we can claim that $\Pi_{CSP}(SecMin)$ is computationally indistinguishable from $\Pi^S_{CSP}(SecMin)$.

On the other hand, the execution image of CP, denoted by $\Pi_{CP}(SecMin)$, is given by $\Pi_{CP}(SecMin) = \{[u]_{pk}, [S]_{pk}\}$. Let the simulated image of CP be given by $\Pi^S_{CP}(SecMin) = \{\alpha, \beta\}$, where α and β are randomly chosen from $\mathbb{Z}_N$. Since DT-PKC is a semantically secure encryption scheme, $[u]_{pk}$ and $[S]_{pk}$ are computationally indistinguishable from α and β, respectively. Thus, we can say that $\Pi_{CP}(SecMin)$ is computationally indistinguishable from $\Pi^S_{CP}(SecMin)$. Based on the above analysis, we can claim that **SecMin** protocol is sure under the semi-honest model.

The security proof of $\mathbf{SecMI}_n$ and **SecComp** protocols are similar to that of **SecDist** and **SecMin** protocols under the semi-honest model.

6.2 Security of Sec$k$NN Scheme

In a similar manner, we can prove that our Sec$k$NN scheme is secure under the semi-honest model firstly. Briefly, during Step 1 to Step 3 in Algorithm 4, the sub-protocols **SecDict** and $\mathbf{SecMI}_n$ are used as the fundamental building block, the rest of data process is non-interactive. Therefore, this phase is secure. Furthermore, the data operations in Step 4 - Step 7 are similar to the process of **SecDist**, all the exchanged messages are in encrypted format and each value deduced by CP and CSP is blinded by random numbers. As a result, it is easy to prove that Sec$k$NN scheme is secure under the semi-honest model.

Next we discuss our Sec$k$NN scheme can preserve the data confidentiality and query privacy against an active adversary $\mathcal{A}$. If $\mathcal{A}$ eavesdrop the transmission link between DO and CP, the encrypted database are got by $\mathcal{A}$. Moreover, all the intermediate values transmitted between CP and CSP may also be eavesdroped by $\mathcal{A}$. Because all these data are transmitted in encrypted form or randomized by the random numbers involved in the protocols, $\mathcal{A}$ cannot decrypt the ciphertext and intermediate values unless $\mathcal{A}$ obtains the strong private key. However, it is impossible to recover the strong private key for $\mathcal{A}$, as the private key is randomly split by executing **SkeyS** algorithm of DT-PKC and $\mathcal{A}$ cannot capture both shares. Furthermore, even $\mathcal{A}$ compromises some of other QUs (i.e. not the QU who launched the query) to get their weak private, $\mathcal{A}$ is still unable to decrypt the above ciphertext or the encrypted final result due to the unrelated property of different QUs' weak private keys in DT-PKC. In all, the data confidentiality and query privacy which defined in Section 4.2 was satisfied.

7 Performance Evaluation

In this section, we developed a Java prototype that implements our scheme with the performance improvement strategies, and the evaluations were performed on a machine with 3.30GHz four-cores processor and 8GB RAM. Specifically, we implement DT-PKC cryptosystem by BigInteger Class in java development kit, and using this implement our computation protocols. For our experiments on real dataset, we used the gas sensor array under dynamic gas mixtures dataset [48] that consists of 4,178,504 data records and 19 attributes (i.e. dimensions). To make a comprehensive performance evaluation, we compare our query processing scheme with the SkNN$_b$ [10] which adopts the Twins-Cloud structure and Paillier cryptosystem. In addition, it has lower security than ours. Moreover, we do the performance analysis of both schemes by varying parameters.

We test our scheme over above real-world dataset with different scales (i.e. n from $200,000$ to 1 million). Regardless of n, the default cardinality of each rectangular partition (b) is set to 200 in the k-d tree space-partitioning. For the DT-PKC algorithm, we denote N as 1024 bits to achieve 80-bit security levels [49]. At least 20 random k-NN queries are selected and evaluated with each scale. Table 3 presents the specific parameter settings in our experiment. In all experiments, unless otherwise stated, when we consider just one parameter, we keep all other parameters at their default values. And the completion of our scheme is accompanied by the optimization of performance mentioned in Section 5.3. First of all, we evaluate the proposed scheme with the main performance metrics, including data processing time at the data owner, k-NN query response time and computational cost at the query users, and secondly, we evaluate the scalability of our system, including two factors: the number of the query users and whether the scheme uses optimized methods.

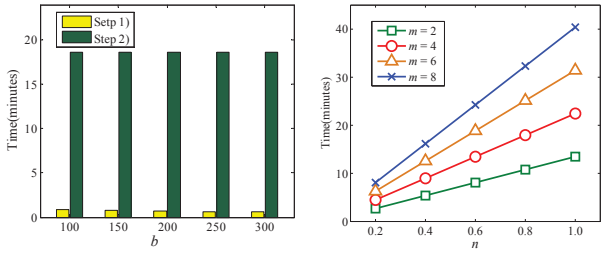
7.1 Data Processing Time at the Data Owner

In the phase of data pretreatment, there are two major steps for the data owner: 1) creating k-d tree space index and 2) encrypting data. The data processing time of our scheme mainly depends on three parameters: 1) the size of dataset (n), 2) the cardinality of each rectangular partition (b), and 3) the number of attributes (m). Therefore, we evaluate the performance by varying these three parameters.

TABLE 3
Parameter Settings

Parameter	Values	Default Value
Dataset size ($n/\times 10^5$)	$0.2, 0.4, 0.6, 0.8, 1.0$	0.6
Cardinality of partition (b)	$100, 200, 300, 400$	200
Required k (k)	$2, 4, 6, 8$	6
Number of attributes (m)	$2, 4, 6, 8$	6
Length of N (l / bit)	1024	1024

For $n = 0.6$ and $m = 6$, Fig. 4a shows the data processing time of our scheme for varying values of b. The experimental results show that the time in Step 2) takes the majority of total time and does not change at all with varying b. This is because that most of the cost comes from the data encryption which is independent of b. For example, when $b = 200$, the time of Step 2) is 18.67 minutes whereas Step 1) only requires 0.69 minutes. As is shown in Fig. 4b, it is natural to see a linear increase with respective to n and m in the total data processing time. This is explained by the fact that the time of both steps is mainly depends on the size of dataset (n) and the number of attributes (m).



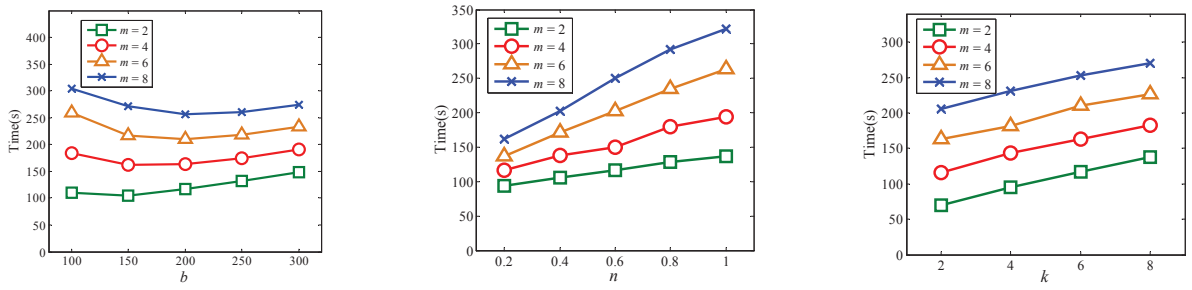
(a) The time of each step for $n = 0.6$ and $m = 600$ (b) The time for $b = 200$

Fig. 4. Data processing time by varying parameters in our scheme.

Next, we plot the data processing time of SkNN$_b$ and Sec$k$NN by varying n and m in Fig. 5. As the n and m increases, the time of two schemes linearly increases because they both need to encrypt all the data. It is also obvious that, the time of SkNN$_b$ is less than ours. For example, in Fig. 5a, when the dataset has $60,000$ records, the time of SkNN$_b$ is 9.98 minutes while ours needs 19.36 minutes. The main reason is that the single encrypting time of DT-PKC is more than that of Paillier cryptosystem. However, our scheme still enjoys practical performance on massive datasets for data processing at the data owner shown in Fig. 5. Even more important, this is only a one-time cost.

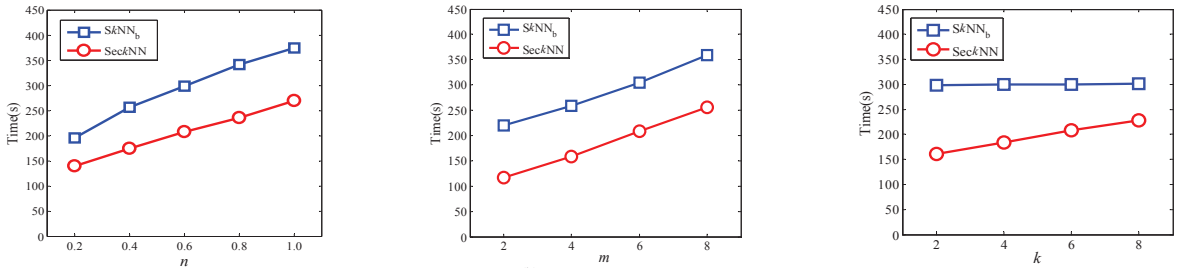
7.2 k-NN Query Response Time

The main performance metric used to evaluate the proposed technique is k-NN query processing time. Similar to [29], We define the indicator as the duration from the time the query is issued until the k results are received at the query users. For our scheme, this time consists of two parts: 1) figure out



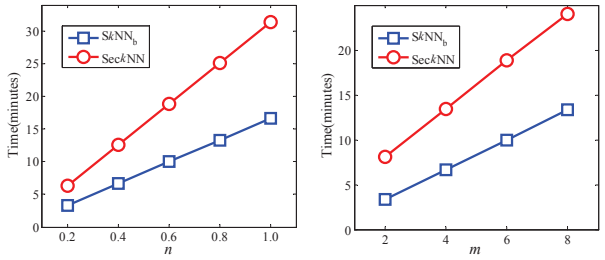
(a) $n = 0.6$ and $k = 6$ (b) $b = 200$ and $k = 6$ (c) $n = 0.6$ and $b = 200$

Fig. 6. Query response time by varying parameters in our scheme.



(a) $m = 6$, $k = 6$ and $b = 200$ (b) $n = 0.6$, $k = 6$ and $b = 200$ (c) $n = 0.6$, $m = 6$ and $b = 200$

Fig. 7. Query response time of SkNN$_b$ and Sec$k$NN.



(a) $m = 6$ and $b = 200$ (b) $n = 0.6$ and $b = 200$

Fig. 5. Data processing time of SkNN$_b$ and Sec$k$NN.

the partition that contains a query point q, which is mainly affected by the total number of rectangular partitions; 2) find k nearest neighbor of q among points in this partition element through a secure way.

Firstly we analyze the performance of our scheme by varying parameters. Fig. 6a shows the query response time by varying b and m. When $m = 2$ and $m = 4$ ($m = 6$ and $m = 8$), the query response time is decreased as b changes from 100 to 150 (from 100 to 200) while the query response time increase as b changes from 150 to 300 (from 200 to 300). This result comes from the following reasons. The total number of rectangular partitions decreases as b becomes larger. So, with b increases, the less computation time is required for **SecComp** algorithm to find a partition corresponding to the query. However, the more larger cardinality of the rectangular cell lead more overhand to perform the **Sec$k$NN** algorithm in a partition. The k-NN query response time for $b = 200$ and $k = 6$ varying values of n and m are shown in Fig. 6b. The observation is that the time grows linearly with n and m. When $n = 0.6$ and $b = 200$, a similar trend is observed for varying values of k and m, which is observed in Fig. 6c. Combining the two conditions, it is clear that the response time of our scheme is linear with with k, m and n.

Next, Fig. 7 shows the query response time of SkNN$_b$ and Sec$k$NN by varying parameters. Note that we also use offline and pipeline computation to improve efficiency for SkNN$_b$. The performance of both schemes by varying n is shown in Fig. 7a. As the n increases, the query processing time of SkNN$_b$ linearly grows because it needs to perform secure protocols over all the data. For example, the time of SkNN$_b$ varies from 198 to 380 seconds when n ($\times 10^5$) is changed from 0.2 to 1.0 respectively. Also, for Sec$k$NN, the time varies from 140 to 271 seconds when n is changed from 0.2 to 1.0 respectively. That is, the time cost of Sec$k$NN is more than 40% less than SkNN$_b$, which benefit from the k-d tree space-partitioning. As shown in Fig. 7b, a similar trend is observed for varying values of m, because the dimension m also affects each step of solutions. We evaluated the response time of both scheme for varying k, the results are shown in Fig. 7c. For Sec$k$NN, the time grows linearly with k, but for SkNN$_b$, the time does not change much with varying k. This is because the computation for top-k minimum values is in plaintext form, which is done at the expense of security. Above all, our scheme assures data security and user privacy while decreasing as much computing time as possible.

7.3 Computational Cost at the Query User

Fig. 8 shows the computational cost at the query users of SkNN$_b$ and Sec$k$NN for varying the k and m. As the k becomes larger, the cost time of both schemes linearly increase. Meanwhile, when the m increases, the cost time of both schemes also increases accordingly, because the schemes

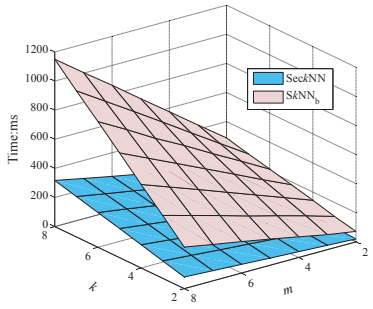


Fig. 8. Computational cost at the query users of SkNN$_b$ and Sec$k$NN.

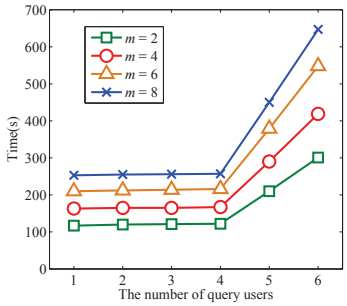


Fig. 9. Query response time by varying numbers of query users in our scheme.

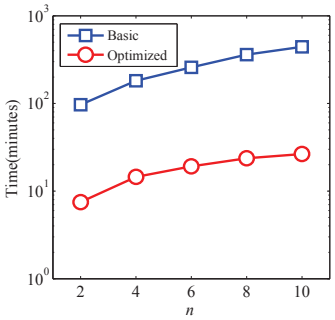


Fig. 10. Comparison of query response time between basic and optimized scheme.

should perform $k \cdot m$ times decryption. As we can observe, the computational cost of SkNN$_b$ is always much than ours for all cases. For instance, when $k = 6$ and $m = 6$, the time of SkNN$_b$ is 648 ms while ours is 180 ms. This is due to the difference of decryption cost between Paillier and DT-PKC cryptosystem (i.e. The cost of **WDec** algorithm is less than decryption in Paillier under similar secure conditions). In short, our scheme has low computation cost at the query users, which is suitable for the lightweight client.

7.4 Scalability

In this part, we further explore the effects of the number of the query users and optimization techniques. In our scheme, the number of the query users can be understood as the query concurrency. Fig. 9 shows the impact of changing the number of query users on query performance. For all the values of m, query response time is nearly invariable as the number of query users (concurrency) changes from 1 to 4 while the query response time increases linearly as the number is greater than 4. Since multithread programming technique was used to improve query efficiency in our Java prototype and the resource that every thread competes for is only computing power of CPU, the boost in concurrency only increases the CPU utilization without query response time degradation, when the concurrency goes below a certain threshold value. Once above this threshold, an increase in the number of concurrent visitors causes the waiting for CPU with it came an approximately liner increase in query response time. As shown in Fig. 9, the best value to pick for the concurrency value (i.e. the threshold value) in our implementation is the number of CPUs on the machine. Note that Hadoop/Spark provides a framework to improve concurrency by distributing the workload into a cluster of computers, such technologies are beyond the scope of this work, We will leave this to future work.

As mentioned earlier, high performance can be achieved with the optimization techniques (k-d tree and the offline and pipeline computing). To further justify this claim, we implement a comparison of query response time between basic and optimized scheme, the result is as shown in Fig. 10. With optimization techniques, the time for query is reduced by orders of magnitude and faster-than-line. It is because that k-d tree can dramatically trim the database size and offline and pipeline computing can reduce the number of waiting threads in the execution of the program. Therefore, it is necessary and effective to optimize our scheme through these techniques.

7.5 Further Discussions

In above-mentioned experiments, we focus on the evaluation of computational cost, not on the communication costs. This is because the amount of computations can be controlled but the communication rate is uncontrollable in the practical Twin-Cloud environment. We also acknowledge that it will take a longer time to execute query in practical cloud environment than that in our simulation experiment. The communication latency will be the focus in the further work.

8 Conclusion

In this paper, we focused on the problem of supporting k-NN query over encrypted cloud data while the data owner cannot share his key with query users. For this we proposed a new solution with mutiple keys to solve the key-sharing problems thoroughly. At the core of our scheme, we presented a series of novel secure protocols based on Twin-Cloud structure and DT-PKC cryptosystem. We showed a theoretical analysis that our scheme can protect the data confidentiality and query privacy. Finally, extensive experimental evaluations demonstrate the efficiency and the scalability of the our scheme. As a future work, we will extend our work to support other data mining tasks, such as classification and similarity computation.

Acknowledgments

The corresponding author is Yulong Shen. This paper is supported in part by the National Natural Science Foundation of China (under the grant No. U1536202, 61571352, 61373173, 61472001, 61602364, 61602365, 61572001 and

61472001), as well as the Fundamental Research Funds for the Central Universities (under the grant No. BDY131419, XJS16042, XJJS15002, JB160312 and JBG160303).

REFERENCES

[1] Z. Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, "Deduplication on encrypted big data in cloud," *IEEE Transactions on Big Data*, vol. 2, no. 2, pp. 138–150, 2016.

[2] S. Yu, Y. Tian, S. Guo, and D. O. Wu, "Can we beat ddos attacks in clouds?" *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2245–2254, Sept 2014.

[3] S. Yu, S. Guo, and I. Stojmenovic, "Can we beat legitimate cyber behavior mimicking attacks from botnets?" in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 2851–2855.

[4] M. Li, S. Yu, W. Lou, and Y. T. Hou, "Toward privacy-assured cloud data services with flexible search functionalities," in *2012 32nd International Conference on Distributed Computing Systems Workshops*. IEEE, 2012, pp. 466–470.

[5] H. Cui, X. Yuan, and C. Wang, "Harnessing encrypted data in cloud for secure and efficient image sharing from mobile devices," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2659–2667.

[6] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Privacy-preserving query over encrypted graph-structured data in cloud computing," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. IEEE, 2011, pp. 393–402.

[7] E. Kabir, A. Mahmood, H. Wang, and A. Mustafa, "Microaggregation sorting framework for k-anonymity statistical disclosure control in cloud computing," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[8] 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.

[9] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 733–744.

[10] 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.

[11] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider, "Twin clouds: An architecture for secure cloud computing," in *Workshop on Cryptography and Security in Clouds (WCSC 2011)*, 2011.

[12] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[13] H. Löhr, A.-R. Sadeghi, and M. Winandy, "Securing the e-health cloud," in *Proceedings of the 1st ACM International Health Informatics Symposium*. ACM, 2010, pp. 220–229.

[14] J. Zhou, Z. Cao, X. Dong, and X. Lin, "Tr-mabe: White-box traceable and revocable multi-authority attribute-based encryption and its applications to multi-level privacy-preserving e-healthcare cloud computing systems," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2398–2406.

[15] Y. Zhu, Z. Huang, and T. Takagi, "Secure and controllable k-nn query over encrypted cloud data with key confidentiality," *Journal of Parallel and Distributed Computing*, vol. 89, pp. 1–12, 2016.

[16] Y. Zhu, Z. Wang, and Y. Zhang, "Secure k-nn query on encrypted cloud data with limited key-disclosure and offline data owner," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2016, pp. 401–414.

[17] K. Cheng, L. Wang, and H. Zhong, "Secure nearest neighbor query on crowd-sensing data," *Sensors*, vol. 16, no. 10, p. 1545, 2016.

[18] X. Liu, R. H. Deng, K. K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[19] O. Goldreich, "The foundations of cryptography, vol. 2, chapter general cryptographic protocols," *Cambridge U*, 2004.

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

[21] W. K. Wong, B. Kao, D. W. L. Cheung, R. Li, and S. M. Yiu, "Secure query processing with data interoperability in a cloud database environment," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 1395–1406.

[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] W. Song, B. Wang, Q. Wang, Z. Peng, W. Lou, and Y. Cui, "A privacy-preserved full-text retrieval algorithm over encrypted data for cloud storage applications," *Journal of Parallel and Distributed Computing*, vol. 99, pp. 14–27, 2017.

[24] B. Wang, M. Li, H. Wang, and H. Li, "Circular range search on encrypted spatial data," in *Communications and Network Security (CNS), 2015 IEEE Conference on*. IEEE, 2015, pp. 182–190.

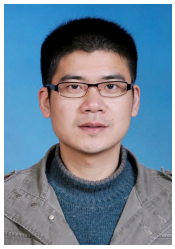
[25] P. Wang and C. V. Ravishankar, "Secure and efficient range queries on outsourced databases using rp-trees," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 314–325.

[26] J. Vaidya and C. Clifton, "Privacy-preserving top-k queries," in *21st International Conference on Data Engineering (ICDE'05)*. IEEE, 2005, pp. 545–546.

[27] S. Liu, Q. Qu, L. Chen, and L. M. Ni, "Smc: A practical schema for privacy-preserved data sharing over distributed data streams," *IEEE Transactions on Big Data*,

vol. 1, no. 2, pp. 68–81, 2015.
[28] J. Yu, P. Lu, Y. Zhu, G. Xue, and M. Li, "Toward secure multikeyword top-k retrieval over encrypted cloud data," *IEEE transactions on dependable and secure computing*, vol. 10, no. 4, pp. 239–250, 2013.
[29] S. Choi, G. Ghinita, H.-S. Lim, and E. Bertino, "Secure knn query processing in untrusted cloud environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 11, pp. 2818–2831, 2014.
[30] B. Wang, Y. Hou, and M. Li, "Practical and secure nearest neighbor search on encrypted large-scale data," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.
[31] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill, "Order-preserving symmetric encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2009, pp. 224–241.
[32] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 463–477.
[33] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
[34] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2003, pp. 37–54.
[35] F. P. Preparata and M. Shamos, *Computational geometry: an introduction*. Springer Science & Business Media, 2012.
[36] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.
[37] Q. Jiang, J. Ma, and F. Wei, "On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–4, 2016.
[38] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–1, 2016.
[39] D. Wang, N. Wang, P. Wang, and S. Qing, "Preserving privacy for free: efficient and provably secure two-factor authentication scheme with user anonymity," *Information Sciences*, vol. 321, pp. 162–178, 2015.
[40] H. Wang, J. Cao, and Y. Zhang, "A flexible payment scheme and its role-based access control," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 425–436, March 2005.
[41] M. Li, X. Sun, H. Wang, Y. Zhang, and J. Zhang, "Privacy-aware access control with trust management in web service," *World Wide Web*, vol. 14, no. 4, pp. 407–430, 2011.
[42] C. Gentry, "Fully homomorphic encryption using ideal lattices," *Proceedings of the Annual Acm Symposium on Theory of Computing*, vol. 9, no. 4, pp. 169–178, 2009.
[43] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *International Workshop on Public Key Cryptography*. Springer, 2010, pp. 420–443.
[44] M. Clear and C. McGoldrick, "Multi-identity and multi-key leveled fhe from learning with errors," in *Annual Cryptology Conference*. Springer, 2015, pp. 630–656.
[45] M. Van Dijk and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing." *HotSec*, vol. 10, pp. 1–8, 2010.
[46] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 965–981, 1998.
[47] M. J. Quinn, *Parallel Programming*. TMH CSE, 2003, vol. 526.
[48] J. Fonollosa, S. Sheik, R. Huerta, and S. Marco, "Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring," *Sensors and Actuators B: Chemical*, vol. 215, pp. 618–629, 2015.
[49] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Nist special publication 800-57," *NIST Special Publication*, vol. 800, no. 57, pp. 1–142, 2007.

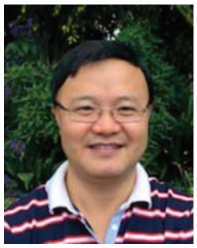
Ke Cheng received his B.S. degree in computer science in Anhui University, Hefei, China, in 2015. He is working toward the MS degree with the school of computer science and technology, Auhui University. And he is currently the visiting student in the school of computer science and technology, Xidian University. His research interests include data security and privacy protection technology.



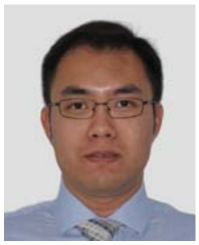
Liangmin Wang received his B.S. degree in Computational Mathematics in Jilin University, Changchun, China, in 1999, and the Ph.D degree in Cryptology from Xidian University, Xi'an, China, in 2007. He is a full professor in the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China. He has been honored as a "Wan-Jiang Scholar" of Anhui Province since Nov. 2013. Now his research interests include data Security & Privacy.

He has published over 60 technical papers at premium international journals and conferences, like IEEE TITS, IEEE Transactions on Vehicular Technology, IEEE GlobeCOM, IEEE WCNC. Dr WANG has been severed as the TPC of many IEEE conferences, such as IEEE ICC, IEEE HPCC, IEEE TrustCOM. Now he is an associate editor of Security and Communication Networks, a member of IEEE, ACM, and a senior member of Chinese Computer Federation.

Yulong Shen received the B.S. and M.S. degrees in Computer Science and Ph.D. degree in Cryptography from Xidian University, Xi'an, China, in 2002, 2005, and 2008, respectively. He is currently a Professor at the School of Computer Science and Technology, Xidian University, China. He is also an associate director of the Shaanxi Key Laboratory of Network and System Security and a member of the State Key Laboratory of Integrated Services networks Xidian University, China. He has also served on the technical program committees of several international conferences, including ICEBE, INCoS, CIS and SOWN. His research interests include Wireless network security and cloud computing security.



Hua Wang received his PhD degree from the University of Southern Queensland, Australia. He is now a full time Professor at Victoria University. He was a professor at the University of Southern Queensland before he joined Victoria University. Hua has more than ten years teaching and working experience in Applied Informatics at both enterprise and university. He has expertise in electronic commerce, business process modeling and enterprise architecture. As an Chief Investigator, three Australian Research Council (ARC) Discovery grants have been awarded since 2006, and 155 peer reviewed scholar papers have been published. Six PhD students have already graduated under his principal supervision.



Yongzhi Wang received his BS and MS degrees in computer science from Xidian University, China in 2004 and 2007, respectively. He received his PhD in computer science from Florida International University, FL, USA, in 2015. Since August 2015, he has been an Assistant Professor at Xidian University, China. His research interests include big data, cloud computing security and outsourced computing security.

Xiaohong Jiang received his B.S., M.S. and Ph.D degrees in 1989, 1992, and 1999 respectively, all from Xidian University, China. He is currently a full professor of Future University Hakodate, Japan. Before joining Future University, Dr.Jiang was an Associate professor, Tohoku University, from Feb.2005 to Mar.2010. Dr. Jiangs research interests include computer communications networks, mainly wireless networks and optical networks, network security, routers/switches design, etc. He has published over 260 technical papers at premium international journals and conferences, which include over 50 papers published in top IEEE journals and top IEEE conferences, like IEEE/ACM Transactions on Networking, IEEE Journal of Selected Areas on Communications, IEEE Transactions on Parallel and Distributed Systems, IEEE INFOCOM. Dr. Jiang was the winner of the Best Paper Award of IEEE HPCC 2014, IEEE WCNC 2012, IEEE WCNC 2008, IEEE ICC 2005-Optical Networking Symposium, and IEEE/IEICE HPSR 2002. He is a Senior Member of IEEE, a Member of ACM and IEICE.

Hong Zhong received her B. S. degree in applied mathematics in Anhui University, China, in 1986, and the Ph.D degree in computer science and technology from University of Science and Technology of China (USTC), China, in 2005. Now she is a professor and Phd Advisor of Anhui University. Her research interests include security protocols and wireless sensor networks.