J. Parallel Distrib. Comput. 89 (2016) 1–12

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Secure and controllable k-NN query over encrypted cloud data with key confidentiality

Youwen Zhu [a,b,*], Zhiqiu Huang [a], Tsuyoshi Takagi [b]

[a] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China
[b] Institute of Mathematics for Industry, Kyushu University, Fukuoka, 819-0395, Japan

HIGHLIGHTS

- We present a new scheme for encrypting the outsourced database and query points.
- The new scheme can effectively support k-nearest neighbor computation while preserving data privacy and query privacy.
- The new scheme enables data owner to keep his key in private, instead of sharing the key with query users.
- In the new scheme, query users cannot launch any feasible k-nearest neighbor query without approval of data owner.
- Experimental results validate the efficiency of the new approach.

ARTICLE INFO

Article history:
Received 28 May 2014
Received in revised form
25 August 2015
Accepted 30 November 2015
Available online 12 December 2015

Keywords:
Cloud computing
Privacy
k-nearest neighbors
Query

ABSTRACT

To enjoy the advantages of cloud service while preserving security and privacy, huge data are increasingly outsourced to cloud in encrypted form. Unfortunately, most conventional encryption schemes cannot smoothly support encrypted data analysis and processing. As a significant topic, several schemes have been recently proposed to securely compute k-nearest neighbors (k-NN) on encrypted data being outsourced to cloud server (CS). However, most existing k-NN search methods assume query users (QUs) are fully-trusted and know the key of data owner (DO) to encrypt/decrypt outsourced database. It is not realistic in many situations.

In this paper, we propose a new secure k-NN query scheme on encrypted cloud data. Our approach simultaneously achieves: (1) *data privacy* against CS: the encrypted database can resist potential attacks of CS, (2) *key confidentiality* against QUs: to avoid the problems caused by key-sharing, QUs cannot learn DO's key, (3) *query privacy* against CS and DO: the privacy of query points is preserved as well, (4) *query controllability*: QUs cannot launch a feasible k-NN query for any new point without approval of DO. We provide theoretical guarantees for security and privacy properties, and show the efficiency of our scheme through extensive experiments.

© 2015 Elsevier Inc. All rights reserved.

1. Introduction

In recent years, cloud services become more and more prevalent, since they can offer many benefits, such as quick deployment without up-front cost, dynamical allocation and cost reduction. For enjoying the advantages, individuals and organizations are being motivated to centralize their datasets into the convenient pay-as-you-go storage space of cloud service providers, e.g., Amazon, Google, Microsoft. Because the direct physical control will be transferred to cloud server (CS) while outsourcing data to a remote cloud, it arouses the security and privacy concerns. Thus, the sensitive information of outsourced data has to be encrypted by data owner (DO) before they are uploaded to cloud such that no privacy is breached. Meanwhile, DO may want to take advantage of the powerful computation capability of CS to analyze or query the data stored in the cloud for extracting beneficial knowledge and patterns. Nevertheless, encryption will impede the functionality and performance of querying/analyzing over the outsourced dataset in cloud.

As a fundamental query operation, k-nearest neighbors (k-NN) computation [24] aims at finding out k nearest tuples of a given query point according to a certain distance metric, such as

* Corresponding author at: College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China.
E-mail addresses: zhuyouwen@gmail.com, zhuyw@nuaa.edu.cn (Y. Zhu).

http://dx.doi.org/10.1016/j.jpdc.2015.11.004
0743-7315/© 2015 Elsevier Inc. All rights reserved.

Euclidean distance, cosine similarity. For the secure k-NN query on outsourced encrypted data, Wong et al. [35] propose an asymmetric scalar-product-preserving encryption (ASPE) scheme which can achieve better security through using a random invertible matrix to perturb the database than the ones [7] utilizing orthogonal matrix. Because of its nice property in security, ASPE has been applied in many problems [5,6,28]. The works [39,38,36] lately present various schemes to securely support approximate k-NN query. However, in the existing work [35,38,36,39], all query users (QUs) and DO share the key of encryption and decryption which results in that only completely trusted QUs can conduct the query on encrypted cloud data. It will not only limit the scope of applications, but also induce several practical problems. Firstly, since each QU knows the full key, the adversary can obtain the key by corrupting any one QU, which will seriously increase the risk of key leakage. Secondly, in many real-world applications, DO may have no enough trust on QUs. For example, an online social networking service (SNS) site can make use of cloud service to store and manage the data of its members for enjoying the benefit of cloud and saving IT expense. Then, the members may like to query their k-NN members according to some social proximity. Under the framework of [35,38,36,39], the members have to encrypt their attributes with the key which is almost the same as the one that the SNS site encrypts and decrypts his outsourced dataset. It is not realistic, as the site cannot share the key with each member otherwise it will severely violate the business secret and other members' privacy. Last but not the least, once a QU receives the key, her query processing will not be controlled by DO any more, and it is difficult to revoke the key distributed to the QU even she is deemed to be untrustworthy. In general, the existing secure k-NN query schemes where QUs can access the key of DO are still far from being practical in many situations. The previous work [42,43] considers the risk of key leakage from QUs, but the QUs in [42,43] still can learn partial sum of the numbers in the key of DO and upon receiving the partial sum through one legal query, the QUs also can launch uncontrolled queries. Yuan et al. [40] present a secure NN ($k = 1$) query scheme to resist untrusted query clients and CS, but the query clients in [40] directly submit private plain query vectors to DO, thus, their scheme does not preserve the query privacy. Besides, Zhu et al. [41] recently prove that Yuan's scheme [40] cannot achieve their declared security, and the encrypted dataset in [40] can be fast broken by untrusted QUs and CS.

In this paper, we focus on secure k-NN query over encrypted cloud data, and propose a new efficient encryption and secure query scheme to address the above problems. With a novel combination of random matrix transformation, random permutation, additively homomorphic encryption [21] and dimension extension etc., our proposed approach can simultaneously achieve the following security and privacy properties in efficiency. (1) *Data privacy* against CS: DO encrypts his private database and outsources the encrypted dataset to CS for using the storage and computation ability of the cloud while preserving the privacy of outsourced data. (2) *Key confidentiality* against QUs: to reduce the risk of key leakage and respond other drawbacks of key-sharing, DO will keep his key confidential and no QU can learn it even some QUs collude. (3) *Query privacy* against DO and CS: query points are private to the corresponding QUs throughout the query computation, and no query privacy is breached. (4) *Query controllability*: QUs cannot launch a feasible k-NN query for a new given query point without approval of DO such that the owner can effectively manage the queries on his dataset and prohibit pernicious QUs.

To the best of our knowledge, this work is the first scheme to efficiently support k-NN query on encrypted cloud data with all the four aforementioned security and privacy properties. Our contributions in this paper can be summarized as the following three aspects:

(1) We present a new scheme for encrypting the outsourced database and query points, which can effectively support k-NN computation and preserve data privacy and query privacy. We also propose an improved scheme that can perform secure k-NN query over selectable partial dimensions, if some applications only require a portion of dimensions rather than all of them.
(2) The new approach can provide rigorous guarantees for query controllability and key confidentiality against QUs. Based on the novel properties, the possibility of key leakage will be greatly reduced, and DO can supervise each query on his database.
(3) Through extensive experiments, we evaluate the execution cost of our scheme and compare it with existing work, which shows the efficiency and practicality of our proposed method.

The rest of the paper is organized as follows. In Section 2, we introduce the system model, design goals, and framework of our scheme. Section 3 discusses the existing solutions for k-NN query on encrypted cloud data in detail. Section 4 proposes our new secure k-NN query scheme on encrypted data in cloud, and a further improvement to smoothly support the query over partial dimensions. In Section 5, we provide theoretical guarantees for the security and privacy properties. Section 6 analyzes the overheads of our solution, and utilizes simulation experiments to evaluate our execution cost and compare it with existing approaches. Section 7 reviews the related work. At last, we conclude the paper in Section 8.

2. Problem statement and framework

2.1. Notations

- DO—data owner.
- QU—query user, and QUs denotes query users.
- CS—cloud server.
- $\mathbf{D}$—the private database of DO, consisting of m points $\mathbf{D} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m\}$ in which $\boldsymbol{p}_i = (p_{i1}, p_{i2}, \ldots, p_{id})$.
- $\|\boldsymbol{p}_i\|$—Euclidean norm of $\boldsymbol{p}_i$, i.e., $\|\boldsymbol{p}_i\| = \sqrt{\sum_{j=1}^{d} p_{ij}^2}$.
- $\mathbf{D}'$—the encrypted database of $\mathbf{D}$, being stored in CS, consisting of m points $\mathbf{D}' = \{\boldsymbol{p}'_1, \boldsymbol{p}'_2, \ldots, \boldsymbol{p}'_m\}$ in which $\boldsymbol{p}'_i$ is the encrypted result of $\boldsymbol{p}_i$.
- $\boldsymbol{q}$—a query point, and q_j denotes its jth dimension.
- $\boldsymbol{q}'$—the encrypted result of $\boldsymbol{q}$.
- $\mathcal{D}(\boldsymbol{p}_i, \boldsymbol{q})$—the distance of $\boldsymbol{p}_i$ and $\boldsymbol{q}$, and it equals to $\sum_{j=1}^{d} (p_{ij} - q_j)^2$ where p_{ij} is the jth dimension of $\boldsymbol{p}_i$.
- $\mathbf{0}_{(\epsilon)}$—an ϵ-dimensional zero vector.
- $\boldsymbol{M}_{j*}$—the jth row of matrix $\boldsymbol{M}$.
- $\mathcal{I}_{\boldsymbol{q}}$—index set of k-NN of query point $\boldsymbol{q}$.
- $\boldsymbol{\pi}$—a permutation of several numbers, and $\pi(i)$ denotes the permuted position of ith input, i.e., $\pi(i)$-th number in permuted sequence maps from ith one in original order.
- $\boldsymbol{\pi}^{-1}$—the permutation inverse to $\boldsymbol{\pi}$, i.e., $\pi^{-1}(\pi(i)) = i$.

2.2. System model

In this work, we consider the cloud data storage and outsourcing k-NN query service, involving three kinds of entities: data owner (DO), cloud server (CS) and several query users (QUs), shown in Fig. 1. CS has huge but bounded storage and computing resources. DO privately has a large database $\mathbf{D}$ which consists of m d-dimensional points: $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m$. Each dimension of $\boldsymbol{p}_i$ is a numerical value. Generally, $d \ll m$, for example, d may be less than 100, but m would be much larger than one hundred thousand. To take the advantage of storage resources and computational ability

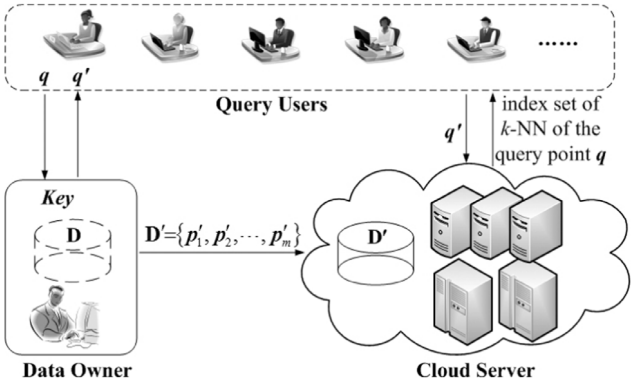


Fig. 1. Architecture of k-NN query on encrypted cloud data.

of cloud service provider, the database $\mathbf{D}$ is outsourced to CS in the encrypted form $\mathbf{D}' = \{\boldsymbol{p}'_1, \boldsymbol{p}'_2, \ldots, \boldsymbol{p}'_m\}$ where $\boldsymbol{p}'_i$ is the encrypted item of $\boldsymbol{p}_i$. Each QU holds several private query points, and every query point $\boldsymbol{q}$ is a d-dimensional point with the same form as $\boldsymbol{p}_i$. The QU wishes to find out the index set of k-NN in $\mathbf{D}$ of her query point, according to the distance $\mathcal{D}(\boldsymbol{p}_i, \boldsymbol{q}) = \sqrt{\sum_{j=1}^{d}(p_{ij} - q_j)^2}$, i.e., the Euclidean distance of $\boldsymbol{p}_i$ and $\boldsymbol{q}$. For privacy concerns, $\boldsymbol{q}$ will be encrypted by the corresponding QU before being submitted to DO and CS. The primary query process is that QU and DO collaboratively compute the encrypted query point $\boldsymbol{q}'$ which is only obtained by the QU, then, QU submits $\boldsymbol{q}'$ to CS who executes an encrypted query on the encrypted database $\mathbf{D}'$ and returns the index set of k-NN of $\boldsymbol{q}$ to the corresponding QU. In this paper, a tuple denotes a point, and we use them interchangeably.

2.3. Threat model and design goals

In this paper, we assume each party is semi-honest, a.k.a., honest-but-curious [13,17], which means each entity will strictly follow the algorithms and return correct computation results, but try to infer as much information of other parties as possible based on the data he receives and holds. Under the model, we aim to efficiently preserve the privacy of DO, query privacy, and query controllability. The details are as follows.

Privacy of DO. To securely protect the privacy of DO, in addition to data privacy, we also consider key confidentiality against QUs.

(1) For **data privacy**, the dataset $\mathbf{D}$ should not be revealed to anybody apart from DO during the outsourcing and k-NN computation. The potential attacker against data privacy is CS who can access the outsourced database. To prevent the attacker from learning the private plain data, only encrypted data will be outsourced and stored in cloud. We consider two security levels for data privacy.

Level-1 attacker only knows the encrypted database and perturbed query points. This corresponds to the ciphertext-only attack in cryptography.

Level-2 attacker is supposed to also know some original tuple in $\mathbf{D}$, except the encrypted database/query points. The level-2 attack is the same as known-sample attack in [19].

(2) Different from data privacy which is used to guarantee the privacy of outsourcing storage, **key confidentiality** means that DO does not disclose his key to QUs during outsourcing k-NN query, i.e., DO can preserve privacy of his key to avert foregoing realistic problems caused by the wide distribution of key. In the existing schemes [35,38,36], DO shares the key with all QUs, then, CS can easily find out all the original tuples of the outsourced database through obtaining the key from one corrupted QU, therefore, they have to assume QUs are fully trusted. In this work, we only assume that QU does not disclose her plain query point to CS, since query point is her privacy and she will rationally give her best protection for it. One of our future work is to deal with malicious QUs who reveal all their data to attacker. Generally, the *attack model* of key confidentiality against QUs is that the attacker (a QU or a collusion of some QUs) learns several query points and the corresponding encrypted results.

Query privacy. The query privacy requires that each query point is privately kept to the corresponding QU during the outsourcing k-NN computation. Neither of DO and CS can learn any plain query point. Similar to the existing works [35,38,36,39,42,43,40], we do not aim to provide the security of query points against the collusion attack of DO and CS, because the coalition of them can inherently infer a query point with high accuracy from its k-NN which is the target output of the query.

Query controllability. It means that any QU, even the QU has obtained the legal encrypted results of several query points, cannot generate a feasible encrypted item for a new given query point without being approved by DO. From the consideration of DO, he can manage the queries in an effective manner, and reject a query on his database if he thinks the QU is suspect. Consequently, the *attacker of query controllability* is also supposed to know several query points and the corresponding legal encrypted results.

Efficiency. Additionally, our scheme will preserve the security and privacy in an efficient and practical manner. That is, the above goals should be achieved with low extra computation cost and communication overheads. The encryption time of DO should be practically low, each QU bears reasonable computation and communication tasks for her query privacy. The search processing of each query should be completed by CS, and computing k-NN on the encrypted database should not notably increase the cost of CS comparing to that in existing efficient approaches.

It should be pointed out that the k-NN relationships among encrypted database points and query points can be observed by CS in our scheme. That is, our solution will leak the data access patterns to CS, which might bring about private information disclosure in some applications. We will cope with the data access pattern leakage problem in future work.

2.4. Framework

Our solution consists of four stages illustrated as follows.

- $\mathsf{KeyGen}(\cdot) \mapsto \{Key\}$: This stage will be completed by only DO. It returns a random secret Key which is the key used to later encrypt tuples in $\mathbf{D}$ and each query point $\boldsymbol{q}$.
- $\mathsf{DBEnc}(\mathbf{D}, Key) \mapsto \{\mathbf{D}'\}$: In this stage, DO will locally encrypt each tuple in $\mathbf{D}$. One point is encrypted as a single unit. The output is the encrypted database $\mathbf{D}' = \{\boldsymbol{p}'_1, \boldsymbol{p}'_2, \ldots, \boldsymbol{p}'_m\}$ which will be sent to CS for storage and on-line applications, such as k-NN query.
- $\mathsf{QueryEnc}(\boldsymbol{q}, Key) \mapsto \{\boldsymbol{q}'\}$: Taking as input the secret Key of DO and QU's private point $\boldsymbol{q}$, this stage enables QU and DO to cooperatively encrypt the query point $\boldsymbol{q}$ such that only QU obtains $\boldsymbol{q}'$, the encrypted result of $\boldsymbol{q}$. Besides, Key will not be disclosed to QU, and DO cannot learn $\boldsymbol{q}$.
- $k\mathsf{NNComp}(\mathbf{D}', \boldsymbol{q}', k) \mapsto \{\mathcal{I}_{\boldsymbol{q}}\}$: Upon receiving an encrypted query $\boldsymbol{q}'$, CS computes the k-NN of the input query point, and returns $\mathcal{I}_{\boldsymbol{q}}$ (the index set of the k-NN in $\mathbf{D}'$) to the corresponding QU.

2.5. Additively homomorphic encryption system

Paillier [21] proposed an efficient cryptosystem with semantic security (IND-CPA). The encryption scheme is additively homomorphic, i.e.,

$$E_{pk}(m_1, r_1) \times E_{pk}(m_2, r_2) = E_{pk}(m_1 + m_2, r_1 r_2),$$
$$E_{pk}(m_1, r_1)^{\kappa} = E_{pk}(\kappa \times m_1, r_1^{\kappa}).$$

Here, E denotes the encryption function, m_1 and m_2 are arbitrary messages in plaintext space, pk is the public key, r_1 and r_2 are random parameters for encryption, $E_{pk}(m_1, r_1)$ denotes the encrypted result of m_1 using the random parameter r_1, and κ is a positive integer. In this paper, we also use $E_{pk}(m_1)$ to denote the encrypted value of m_1 while it is unnecessary to emphasize the random parameter. Paillier cryptosystem is an important secure building block to be used in QueryEnc of our scheme. We concisely describe it as follows. More details of the encryption scheme can be found in [21].

Key generation. Select two large enough primes p and q. Then, the secret key sk is $\lambda = lcm(p-1, q-1)$. The public key pk is (n, g), where $n = pq$ and $g \in \mathbb{Z}^*_{n^2}$ such that $gcd(L(g^\lambda \bmod n^2), n) = 1$. Here, $L(x) = (x-1)/n$.

Encryption. Let m_0 be a number in plaintext space $\mathbb{Z}_n$. Select a random $r \in \mathbb{Z}^*_n$ as the secret parameter, then the ciphertext c_0 of m_0 is $c_0 = g^{m_0} r^n \bmod n^2$.

Decryption. Let $c_0 \in \mathbb{Z}_{n^2}$ be a ciphertext. The plaintext hidden in c_0 is $m_0 = \frac{L(c_0^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$.

3. Discussion about the existing secure k-NN query schemes on encrypted cloud data

Our proposed scheme is an improved approach based on ASPE in [35]. Before introducing the details of our approach, this section will briefly discuss ASPE, and demonstrate that our security and privacy goals cannot be achieved through a plausible improvement. Additionally, this section also discusses the previous scheme in [42] which considers the risk of key leakage from QUs, and it shows that QUs in [42] can breach both key confidentiality and query controllability.

In ASPE [35], the outsourced database is perturbed by a random invertible matrix. Because the distance of tuples in $\mathbf{D}$ is no longer preserved, ASPE can achieve data privacy against level-2 attacks. Let $\boldsymbol{M}$ be a random invertible matrix. Then, ASPE encrypts the database points and query point through the following equation

$$\begin{cases} \boldsymbol{p}'_i = (\boldsymbol{p}, -0.5\|\boldsymbol{p}_i\|^2)\boldsymbol{M}, \\ \boldsymbol{q}' = \boldsymbol{M}^{-1}(\boldsymbol{q}, 1)^T \end{cases}$$

where the additional $(d+1)$-th dimension is used to adapt the Euclidean distance. The work [35] shows that $\sqrt{\|\boldsymbol{p}_i\|^2 - 2\boldsymbol{p}_i \cdot \boldsymbol{q} + \|\boldsymbol{q}\|^2} \geqslant \sqrt{\|\boldsymbol{p}_j\|^2 - 2\boldsymbol{p}_j \cdot \boldsymbol{q} + \|\boldsymbol{q}\|^2}$ if and only if $\|\boldsymbol{p}_i\|^2 - \|\boldsymbol{p}_j\|^2 + 2(\boldsymbol{p}_j - \boldsymbol{p}_i) \cdot \boldsymbol{q} \geqslant 0$ for any $i, j \in [1, m]$. After the encryption of ASPE, they have $\boldsymbol{p}'_i\boldsymbol{q}' = \boldsymbol{p}_i \cdot \boldsymbol{q} - 0.5\|\boldsymbol{p}_i\|^2$, and then $\boldsymbol{p}'_i\boldsymbol{q}' - \boldsymbol{p}'_j\boldsymbol{q}' = (\boldsymbol{p}'_i - \boldsymbol{p}'_j)\boldsymbol{q}' = -0.5\big(\|\boldsymbol{p}_i\|^2 - \|\boldsymbol{p}_j\|^2 + 2(\boldsymbol{p}_j - \boldsymbol{p}_i) \cdot \boldsymbol{q}\big)$. Therefore, CS can infer which one of $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is closer to $\boldsymbol{q}$, by comparing $(\boldsymbol{p}'_i - \boldsymbol{p}'_j)\boldsymbol{q}'$ with 0. That is, ASPE can support k-NN query in encrypted form. Nevertheless, the key $\boldsymbol{M}$ is revealed to all QUs in ASPE, thus it will bring about many realistic problems as we have mentioned in Section 1. The work [35] also presents an enhanced ASPE based on random asymmetric splitting and adding artificial dimensions. However, the enhanced one still assumes QUs are completely trusted, and the splitting configuration and adding artificial dimensions also are known to QUs.

In this paper, we consider a more practical model to simultaneously achieve key confidentiality, query controllability, data privacy, and query privacy. A plausible improved scheme is that DO and QU conduct a secure two-party computation protocol [12,13], taking $\boldsymbol{M}$ and $\boldsymbol{q}$ as input, which enables QU to receive only $\boldsymbol{q}'$ without directly learning $\boldsymbol{M}$ and disclosing $\boldsymbol{q}$. The combination of ASPE and secure two-party computation protocol, however, still cannot achieve query controllability and key confidentiality. The reason is that secure two-party computation protocol only prevents the intermediate data of the protocol from revealing the private inputs, but it does not consider the disclosure from legal outputs. In this situation, the encrypted query $\boldsymbol{q}'$, that is the legal output, will disclose partial information of the key $\boldsymbol{M}$, and QU can deduce the whole key while observing enough encrypted query points. Consider a query set including $(d+1)$ query points $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{d+1}\}$, let the ith column of $\boldsymbol{Q}$ be $(\boldsymbol{q}_i, 1)^T$, $\boldsymbol{Q}' = (\boldsymbol{q}'_1, \boldsymbol{q}'_2, \ldots, \boldsymbol{q}'_{d+1})$ and q_{ij} denote the jth dimension of $\boldsymbol{q}_i$, then, we get $\boldsymbol{Q}' = \boldsymbol{M}^{-1}\boldsymbol{Q}$. By selecting appropriate $\boldsymbol{q}_i$, for example, QU sets $q_{ii} = 1$ and $q_{ij} = 0$ (for any $j \neq i$), $\boldsymbol{Q}$ will be an invertible matrix. Then, the QU can attain $\boldsymbol{M}^{-1} = \boldsymbol{Q}'\boldsymbol{Q}^{-1}$ and further encrypt a new query $\boldsymbol{q}'_{\mathtt{new}} = \boldsymbol{Q}'\boldsymbol{Q}^{-1}(\boldsymbol{q}_{\mathtt{new}}, 1)^T$ without any prior approval of DO. Therefore, it will breach both query controllability and key confidentiality.

The previous work in [42] considers the risk of key leakage from QUs in the outsourcing storage and k-NN computation scene. The work [42] uses the same random vectors to extend each $\boldsymbol{p}_i$ into $(2d+2)$-dimensional vector, and computes the encrypted tuple $\boldsymbol{p}'_i$ through multiplying the extended vector by a $(2d+2) \times (2d+2)$ random invertible matrix $\boldsymbol{M}$. Correspondingly, for encrypting $\boldsymbol{q}$, QU randomly selects a nonce d-dimensional vector $\boldsymbol{t}$, and only sends $(\boldsymbol{q} + \boldsymbol{t})$ to DO, then DO takes as input $\boldsymbol{M}$, $(\boldsymbol{q} + \boldsymbol{t})$ and random parameters, and generates some returns to QU. Based on the returns and $\boldsymbol{t}$, QU can obtain the encrypted item $\boldsymbol{q}'$. At last, CS can linearly compute the k-NN using $\{\boldsymbol{p}'_i : 1 \leqslant i \leqslant m\}$ and $\boldsymbol{q}'$. The scheme can also achieve data privacy against level-2 attack. Nevertheless, the returns to QUs will reveal $(M_{i,2j} - M_{i,2j-1})$ (for all $1 \leqslant i \leqslant (2d+2)$, $1 \leqslant j \leqslant (2d)$), e.g., QUs still learn partial sum of numbers in the key of DO. Its advantage is that even an attacker obtains the partial information of $\boldsymbol{M}$ from some untrustworthy QUs, the scheme in [42] can resist the attacker who also knows only the encrypted points, compared with the complete disclosure in ASPE [35]. Namely, [42] can only attain level-1 security once QUs disclose their knowledge about $\boldsymbol{M}$ to attackers. Moreover, for any new query point $\boldsymbol{q}_{\mathtt{new}}$, QU sets $\boldsymbol{t}_{\mathtt{new}} = (\boldsymbol{q} + \boldsymbol{t}) - \boldsymbol{q}_{\mathtt{new}}$, then she can acquire a feasible encrypted value $\boldsymbol{q}'_{\mathtt{new}}$ based on $\boldsymbol{t}_{\mathtt{new}}$ and the above DO's returns for $\boldsymbol{q}$. Consequently, the approach in [42] can keep neither key confidentiality nor query controllability.

4. Secure k-NN query scheme with query controllability and key confidentiality

This section will present our new approach that can achieve key confidentiality against QUs, query controllability, data privacy, and query privacy.

4.1. Overview of our scheme

In this paper, for efficiently protecting the key from QUs and achieving query controllability, we not only adjust the random invertible matrix transformation, but also, more importantly, perturb the original tuples and query points prior to matrix transformation such that the encrypted query points do not leakage the key of DO. Besides, as the encrypted results in our scheme depend on some nonce random parameters, the same point (database point or query point) will not be encrypted into the same item every time, which also improves the security, but does not reduce the query accuracy at all.

For the convenience of presentation, we introduce our new scheme from two phases. Let $\dot{\boldsymbol{p}}_i$ and $\dot{\boldsymbol{q}}$ denote the output of $\boldsymbol{p}_i$ and $\boldsymbol{q}$ in the first phase, respectively. It is remarkable that we do not directly compute $\dot{\boldsymbol{q}}$ during encrypting query point and nobody knows its values, but it is more easy to illustrate our scheme by using $\dot{\boldsymbol{q}}$. At last, the second phase will finish the encryption, and obtain $\boldsymbol{p}'_i$ and $\boldsymbol{q}'$.

First phase. During computing $\dot{\boldsymbol{p}}_i$, two positive integers c and ϵ will be selected by DO as system security parameters in advance, then, we make use of three perturbation methods, including

tuple-sharing perturbation, dynamic virtual dimensions, and a permutation. In tuple-sharing perturbation, we utilize two random vectors $\boldsymbol{S}$ of $(d+1)$ dimensions and $\boldsymbol{\tau}$ of c dimensions, which are a part of our key and shared by tuples in $\mathbf{D}$. The dynamic virtual dimensions, consisting of ϵ columns, serve as the random parameters for encrypting database points, and they are randomly and independently generated for each point in $\mathbf{D}$. Let the vector $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \ldots, v_{i\epsilon})$ denote ϵ dynamic virtual dimensions of $\boldsymbol{p}_i$. The permutation function $\boldsymbol{\pi}$ takes $(d+1+c+\epsilon)$ numbers as input and returns them in a random sequence. Generally, we have

$$\dot{\boldsymbol{p}}_i = \boldsymbol{\pi}\big(\boldsymbol{S}_{-(d+1)} - 2\boldsymbol{p}_i, S_{d+1} + \|\boldsymbol{p}_i\|^2,\ \boldsymbol{\tau},\ \boldsymbol{v}_i\big), \tag{1}$$

where $\boldsymbol{S}_{-(d+1)}$ denotes $\boldsymbol{S} \setminus \{S_{d+1}\}$, i.e., $(S_1,\ S_2,\ \ldots,\ S_d)$. Correspondingly, in this phase, we perturb $\boldsymbol{q}$ as follows,

$$\dot{\boldsymbol{q}} = \boldsymbol{\pi}\big(\boldsymbol{q},\ 1,\ \boldsymbol{R}^{(\boldsymbol{q})},\ \mathbf{0}_{(\epsilon)}\big), \tag{2}$$

where $\mathbf{0}_{(\epsilon)}$ denotes a zero vector of length ϵ, $\boldsymbol{R}^{(\boldsymbol{q})}$ is a c-dimensional vector as the one-time random parameter for encrypting $\boldsymbol{q}$.

Since the same permutation on two vectors does not change the scalar product of them, then, $\dot{\boldsymbol{p}}_i\dot{\boldsymbol{q}}^T = (\boldsymbol{S}_{-(d+1)} - 2\boldsymbol{p}_i)\boldsymbol{q}^T + (S_{d+1} + \|\boldsymbol{p}_i\|^2) + \boldsymbol{\tau}(\boldsymbol{R}^{(\boldsymbol{q})})^T$. For all $1 \leqslant i, h \leqslant m$, we get

$$\begin{aligned}\dot{\boldsymbol{p}}_i\dot{\boldsymbol{q}}^T - \dot{\boldsymbol{p}}_h\dot{\boldsymbol{q}}^T &= (\|\boldsymbol{p}_i\|^2 - 2\boldsymbol{p}_i\boldsymbol{q}^T) - (\|\boldsymbol{p}_h\|^2 - 2\boldsymbol{p}_h\boldsymbol{q}^T)\\ &= \mathcal{D}(\boldsymbol{p}_i, \boldsymbol{q})^2 - \mathcal{D}(\boldsymbol{p}_h, \boldsymbol{q})^2.\end{aligned} \tag{3}$$

Second phase. After obtaining $\dot{\boldsymbol{p}}_i$, DO can locally compute the encrypted tuple $\boldsymbol{p}'_i = \dot{\boldsymbol{p}}_i\boldsymbol{M}^{-1}$. For each query, DO randomly selects a random positive number $\beta_{\boldsymbol{q}}$, then he and QU collaboratively compute $\boldsymbol{q}' = \beta_{\boldsymbol{q}}\boldsymbol{M}\dot{\boldsymbol{q}}^T$ based on Paillier cryptosystem [21] such that QU receives nothing but $\boldsymbol{q}'$ and DO cannot learn $\boldsymbol{q}$, simultaneously.

Though Paillier cryptosystem can directly encrypt only the integers in $\mathbb{Z}_n$, the scaling and shifting approaches [15,34] can be used to allow non-integer numbers and negative values as input. The scaling procedure converts inputs into integers through multiplying them by 10^{δ} ($\delta > 0$), and scales back the outputs by eliminating the scaling factors. This may cause accuracy loss, but we can choose a properly big δ depending on the application contexts so that accuracy loss is acceptably small. The shifting approach uses $\{(n-1)/2+1, (n-1)/2+2, \ldots, (n-1)\}$ and $\{0, 1, \ldots, (n-1)/2\}$ to denote the negative integers and the non-negative integers, respectively. For simplicity of presentation, we omit the scaling and shifting steps during the formal description of our scheme.

Overall, $\{\boldsymbol{S}, \boldsymbol{\tau}, \boldsymbol{\pi}, \boldsymbol{M}\}$ are the key to encrypt database and query points. Nevertheless, in addition to original points and the determinate key, the encryption procedure also takes as input some one-time random parameters, i.e., $\boldsymbol{v}_i$ for encrypting $\boldsymbol{p}_i$, and $\{\boldsymbol{R}^{(\boldsymbol{q})}, \beta_{\boldsymbol{q}}\}$ for encrypting $\boldsymbol{q}$. Thus, the same point (database point or query point) will be encrypted into different items. Throughout the encryption, QUs cannot learn the key and the nonce $\{\boldsymbol{v}_i, \boldsymbol{R}^{(\boldsymbol{q})}, \beta_{\boldsymbol{q}}\}$. As an intermediate value, $\dot{\boldsymbol{q}}$ is also unknown to QUs.

After our encryption, for any $1 \leqslant i, h \leqslant m$, based on Eq. (3), we have

$$\begin{aligned}\boldsymbol{p}'_i\boldsymbol{q}' - \boldsymbol{p}'_h\boldsymbol{q}' &= \beta_{\boldsymbol{q}}\dot{\boldsymbol{p}}_i\dot{\boldsymbol{q}}^T - \beta_{\boldsymbol{q}}\dot{\boldsymbol{p}}_h\dot{\boldsymbol{q}}^T\\ &= \beta_{\boldsymbol{q}}\Big(\mathcal{D}(\boldsymbol{p}_i, \boldsymbol{q})^2 - \mathcal{D}(\boldsymbol{p}_h, \boldsymbol{q})^2\Big).\end{aligned} \tag{4}$$

Since $\beta_{\boldsymbol{q}}$ is positive, in our scheme, the following result holds

$$\boldsymbol{p}'_i\boldsymbol{q}' \gtreqless \boldsymbol{p}'_h\boldsymbol{q}' \quad \text{if and only if } \mathcal{D}(\boldsymbol{p}_i, \boldsymbol{q}) \gtreqless \mathcal{D}(\boldsymbol{p}_h, \boldsymbol{q}).$$

Therefore, our new approach can correctly and efficiently support the k-NN computation of CS on encrypted database and query points according to the distance $\boldsymbol{p}'_i\boldsymbol{q}'$.

4.2. Formal steps of our scheme

We present the detailed steps of each stage, respectively.

• KeyGen$(\cdot) \mapsto \{Key\}$: DO selects two positive integer c and ϵ as security parameters. Let $\eta = d + c + \epsilon + 1$. He randomly generates an invertible matrix $\boldsymbol{M} \in \mathbb{R}^{\eta\times\eta}$, a $(d+1)$-dimensional vector $\boldsymbol{S} = (S_1, S_2, \ldots, S_{d+1}) \in \mathbb{R}^{d+1}$, a c-dimensional vector $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_c) \in \mathbb{R}^c$, and a permutation function $\boldsymbol{\pi}$ of η numbers. Then, DO sets $Key = \{\boldsymbol{S}, \boldsymbol{\tau}, \boldsymbol{\pi}, \boldsymbol{M}\}$, and keeps the Key in private.

Algorithm 1 Computing $\boldsymbol{A}^{(\boldsymbol{q})}$ based on additively homomorphic Paillier cryptosystem

Input: the permutation $\boldsymbol{\pi}$, the matrix $\boldsymbol{M}$, encrypted dimensions of query point $\{E_{pk}(q_1), E_{pk}(q_2), \ldots, E_{pk}(q_d)\}$, a random positive number $\beta_{\boldsymbol{q}}$, and a c-dimensional random vector $\boldsymbol{R}^{(\boldsymbol{q})}$

Output: η-dimensional encrypted vector $\boldsymbol{A}^{(\boldsymbol{q})}$, the ith dimension of which meets $\mathcal{A}_i^{(\boldsymbol{q})} = E_{pk}\Big(\beta_{\boldsymbol{q}}\boldsymbol{M}_{i*}\dot{\boldsymbol{q}}^T\Big)$. Here, $\boldsymbol{M}_{i*}$ denotes the ith row of matrix $\boldsymbol{M}$, $\eta = d+c+\epsilon+1$, and $\dot{\boldsymbol{q}} = \boldsymbol{\pi}\big(\boldsymbol{q},\ 1,\ \boldsymbol{R}^{(\boldsymbol{q})},\ \boldsymbol{0}_{(\epsilon)}\big)$.

for $i = 1$ to η **do**
  set the initial value $\mathcal{A}_i^{(\boldsymbol{q})} = E_{pk}(0)$,
  for $j = 1$ to η **do**
    $t = \pi^{-1}(j)$,
    if $t < d+1$ **then**
      $\varphi = \beta_{\boldsymbol{q}} \times M_{ij}$,
      $\mathcal{A}_i^{(\boldsymbol{q})} = \mathcal{A}_i^{(\boldsymbol{q})} \times E_{pk}^{\varphi}(q_t)$,
    else if $t = d+1$ **then**
      $\varphi = \beta_{\boldsymbol{q}} \times M_{ij}$,
      $\mathcal{A}_i^{(\boldsymbol{q})} = \mathcal{A}_i^{(\boldsymbol{q})} \times E_{pk}(\varphi)$,
    else if $t \leqslant d+1+c$ **then**
      $\omega = t - d - 1$,
      $\varphi = \beta_{\boldsymbol{q}} \times M_{ij} \times R_{\omega}^{(\boldsymbol{q})}$,
      $\mathcal{A}_i^{(\boldsymbol{q})} = \mathcal{A}_i^{(\boldsymbol{q})} \times E_{pk}(\varphi)$,
    end if
  end for
end for

• DBEnc$(\mathbf{D}, Key) \mapsto \{\mathbf{D}'\}$: For each $\boldsymbol{p}_i \in \mathbf{D}$, DO selects an ϵ-dimensional random vector $\boldsymbol{v}_i \in \mathbb{R}^{\epsilon}$, computes the encrypted item $\boldsymbol{p}'_i = \boldsymbol{\pi}\big(\boldsymbol{S}_{-(d+1)} - 2\boldsymbol{p}_i, S_{d+1} + \|\boldsymbol{p}_i\|^2,\ \boldsymbol{\tau},\ \boldsymbol{v}_i\big)\boldsymbol{M}^{-1}$, and uploads $\boldsymbol{p}'_i$ to CS. Here, $\boldsymbol{S}_{-(d+1)}$ is $\boldsymbol{S} \setminus \{S_{d+1}\}$, i.e., $\boldsymbol{S}_{-(d+1)} = (S_1,\ S_2,\ \ldots,\ S_d)$.

• QueryEnc$(\boldsymbol{q}, Key) \mapsto \{\boldsymbol{q}'\}$: The query encryption stage consists of three steps.

(I) QU generates a public key pair $\{pk, sk\}$ of homomorphic cryptosystem [21]. Let $E_{pk}()$ and $D_{sk}()$ denote the encryption function and decryption function, respectively. Then, the QU encrypts each dimension of private query point $\boldsymbol{q}$ using her public key pk, and sends pk and the ciphertexts $\{E_{pk}(q_1), E_{pk}(q_2), \ldots, E_{pk}(q_d)\}$ to DO.

(II) If DO does not approve the query, he returns $\perp$ to the QU and terminates. Otherwise, DO randomly selects a positive number $\beta_{\boldsymbol{q}} \in \mathbb{R}^+$ and a c-dimensional vector $\boldsymbol{R}^{(\boldsymbol{q})} \in \mathbb{R}^c$, then, he computes η-dimensional encrypted vector $\boldsymbol{\mathcal{A}}^{(\boldsymbol{q})}$ by the way of Algorithm 1, and returns $\boldsymbol{\mathcal{A}}^{(\boldsymbol{q})}$ to QU.

(III) At last, QU decrypts each dimension of the encrypted vector $\boldsymbol{\mathcal{A}}^{(\boldsymbol{q})}$, and obtains the encrypted query $\boldsymbol{q}' = (q'_1, q'_2, \ldots, q'_\eta)^T$ in which $q'_j = D_{sk}\big(\mathcal{A}_j^{(\boldsymbol{q})}\big)$, for all $j \in [1, \eta]$.

• kNNComp$(\mathbf{D}', \boldsymbol{q}', k) \mapsto \{\mathcal{I}_{\boldsymbol{q}}\}$: QU uploads $\boldsymbol{q}'$ to CS. Through linearly scanning $\mathbf{D}'$, CS computes the index set of k-NN of $\boldsymbol{q}'$ according to the distance $\boldsymbol{p}'_i\boldsymbol{q}'$, and sends the index set $\mathcal{I}_{\boldsymbol{q}}$ of the k-NN in $\mathbf{D}'$ to the corresponding QU.

The correctness of Algorithm 1. According to the additively homomorphic property, we have $\mathcal{A}_i^{(\boldsymbol{q})} = E_{pk}\Big(\beta_{\boldsymbol{q}}\boldsymbol{M}_{i*}\dot{\boldsymbol{q}}^T\Big) =$

$E_{pk}\left(\sum_{j=1}^{\eta}\beta_{\boldsymbol{q}}M_{ij}\dot{q}_j\right)=\prod_{j=1}^{\eta}E_{pk}(\beta_{\boldsymbol{q}}M_{ij}\dot{q}_j)$. Algorithm 1 sets the initial value of $\mathcal{A}_i^{(\boldsymbol{q})}$ to $E_{pk}(0)$. Then, DO can compute the required $\mathcal{A}_i^{(\boldsymbol{q})}$ by multiplying each $E_{pk}(\beta_{\boldsymbol{q}}M_{ij}\dot{q}_j)$. For each $j\in[1,\eta]$, let $t=\pi^{-1}(j)$, then, j-th dimension of $\dot{\boldsymbol{q}}$ maps from tth one of $\left(\boldsymbol{q},1,\boldsymbol{R}^{(\boldsymbol{q})},\mathbf{0}_{(\epsilon)}\right)$. Consider $E_{pk}(\beta_{\boldsymbol{q}}M_{ij}\dot{q}_j)$ in the following four situations according to the value of t.

(I) If $1\leqslant t\leqslant d$, we have $\dot{q}_j=q_t$, thus, the ciphertext of $(\beta_{\boldsymbol{q}}M_{ij}\dot{q}_j)$ can be achieved by the exponential computation $E_{pk}(\beta_{\boldsymbol{q}}M_{ij}\dot{q}_j)=E_{pk}^{\beta_{\boldsymbol{q}}M_{ij}}(q_t)$.

(II) While $t=d+1$, then, $\dot{q}_j=1$. Accordingly, DO can locally encrypt $(\beta_{\boldsymbol{q}}M_{ij})$.

(III) If $d+1<t\leqslant d+1+c$, then, $\dot{q}_j=R_{t-d-1}^{(\boldsymbol{q})}$, and the cipher value $E_{pk}\left(\beta_{\boldsymbol{q}}M_{ij}R_{t-d-1}^{(\boldsymbol{q})}\right)$ can also be directly computed by DO.

(IV) At last, when $t>d+1+c$, we have $\dot{q}_j=0$ and it is no need to change the value of $\mathcal{A}_i^{(\boldsymbol{q})}$.

As can be seen, the lines 5–15 in Algorithm 1 correctly deal with the above four aspects. Therefore, DO can obtain the exact $\mathcal{A}_i^{(\boldsymbol{q})}$ after traversing all $j\in[1,\eta]$, which completes the proof of the correctness of Algorithm 1.

A toy example. To vividly show the steps of our scheme, we give a toy example as follows. Consider a simple situation in which DO has a dataset of two points, i.e., $\mathbf{D}=\{\boldsymbol{p}_1,\boldsymbol{p}_2\}$, and QU has one query point $\boldsymbol{q}$. Here each point is 2-dimensional, $\boldsymbol{p}_1=(10,2.7)$, $\boldsymbol{p}_2=(5.3,8.6)$, and $\boldsymbol{q}=(9.5,3.1)$. Suppose DO sets the parameters $c=\epsilon=1$, and QU wants to find the nearest point for each query point, i.e., $k=1$. Then, the four stages of our scheme will be implemented as follows.

• KeyGen$(\cdot)\mapsto\{Key\}$: Since $d=2,c=1,\epsilon=1$, we have $\eta=d+c+\epsilon+1=5$, that is, the encrypted points will be 5-dimensional. Hence, DO randomly selects an invertible 5×5 matrix

$$\boldsymbol{M}=\begin{bmatrix}6.7&1.2&2.6&3.3&5.5\\9.2&45&11&3.2&19\\17&1.5&8.3&2.1&14\\30&2.9&16&20&6.2\\11&28&3.6&23&13\end{bmatrix},$$

a 3-dimensional vector $\boldsymbol{S}=(12,5.0,30)$, a 1-dimensional vector $\boldsymbol{\tau}=(21)$, and a permutation $\boldsymbol{\pi}=(3,1,4,5,2)$. Then, $Key=\{\boldsymbol{M},\boldsymbol{S},\boldsymbol{\tau},\boldsymbol{\pi}\}$, and DO keeps Key in private.

• DBEnc$(\mathbf{D},Key)\mapsto\{\mathbf{D}'\}$: Our scheme encrypts the dataset $\mathbf{D}$ point by point. Here, we give the details of encrypting $\boldsymbol{p}_1$. Since $\epsilon=1$, DO generates a 1-dimensional vector $\boldsymbol{v}_1=(8.0)$ and computes the inverse of the matrix $\boldsymbol{M}$,

$$\boldsymbol{M}^{-1}=\begin{bmatrix}584.194544&38.15879&-214.156398&6.940024&-75.609659\\186.15253&12.182895&-68.272185&2.211715&-24.093536\\-722.312634&-47.148219&264.769994&-8.518099&93.428304\\-243.412824&-15.923013&89.223565&-2.879786&31.540965\\-264.583875&-17.300373&97.079043&-3.182157&34.272397\end{bmatrix}.$$

Then, DO computes the encrypted result of $\boldsymbol{p}_1$ as follows.

$$\begin{aligned}\boldsymbol{p}_1'&=\boldsymbol{\pi}\left(\boldsymbol{S}_{-(d+1)}-2\boldsymbol{p}_i,S_{d+1}+\|\boldsymbol{p}_i\|^2,\ \boldsymbol{\tau},\ \boldsymbol{v}_i\right)\boldsymbol{M}^{-1}\\&=\boldsymbol{\pi}\left(12-2*10,5.0-2*2.7,30+10^2+2.7^2,21,8.0\right)\boldsymbol{M}^{-1}\\&=\boldsymbol{\pi}\left(-8.0,-0.4,137.29,21,8.0\right)\boldsymbol{M}^{-1}\\&=\left(137.29,-8.0,21,8.0,-0.4\right)\boldsymbol{M}^{-1}\\&=(61704.81,4030.78,-22620.23,734.46,-7987.09).\end{aligned}$$

Similarly, if randomly selecting $\boldsymbol{v}_2=(17)$, we have

$$\boldsymbol{p}_2'=(60001.92,3919.68,-21996.93,714.60,-7766.56).$$

• QueryEnc$(\boldsymbol{q},Key)\mapsto\{\boldsymbol{q}'\}$: As Paillier homomorphic encryption system can directly encrypt only integers, QU and DO need to transform their data into integers by multiplying scale factors before the query encryption. Here, we assume the scale factors of both QU and DO are 10. Then, the three steps of this stage are as follows.

Firstly, QU generates a public key pair $\{pk,sk\}$ of Paillier cryptosystem, and sends $E_{pk}(10*\boldsymbol{q})=\left(E_{pk}(95),E_{pk}(31)\right)$ to DO, where 10 is the scale factor.

Secondly, as $c=1$, DO creates a 1-dimensional random vector $\boldsymbol{R}^{(\boldsymbol{q})}=(11)$, and computes

$$\begin{aligned}&\boldsymbol{\pi}\left(E_{pk}(10*\boldsymbol{q}),E_{pk}(10*1),E_{pk}(10*\boldsymbol{R}^{(\boldsymbol{q})}),E_{pk}(10*0)\right)\\&=\boldsymbol{\pi}\left(E_{pk}(95),E_{pk}(31),E_{pk}(10),E_{pk}(110),E_{pk}(0)\right)\\&=\left(E_{pk}(10),E_{pk}(95),E_{pk}(110),E_{pk}(0),E_{pk}(31)\right).\end{aligned}$$

Then, DO selects a random positive number $\beta_{\boldsymbol{q}}=3.0$, and computes the 5-dimensional vector $\boldsymbol{\mathcal{A}}^{(\boldsymbol{q})}=(\mathcal{A}_1^{(\boldsymbol{q})},\mathcal{A}_2^{(\boldsymbol{q})},\ldots,\mathcal{A}_5^{(\boldsymbol{q})})$. While computing $\mathcal{A}_1^{(\boldsymbol{q})}$, DO will calculate $10*\beta_{\boldsymbol{q}}*(M_{11},M_{12},M_{13},M_{14},M_{15})=(201,36,78,99,165)$, and further obtain

$$\mathcal{A}_1^{(\boldsymbol{q})}=E_{pk}(10)^{201}*E_{pk}(95)^{36}*E_{pk}(110)^{78}*E_{pk}(0)^{99}*E_{pk}(31)^{165}.$$

The other dimensions can be calculated by the similar steps.

$$\begin{aligned}\mathcal{A}_2^{(\boldsymbol{q})}&=E_{pk}(10)^{276}*E_{pk}(95)^{1350}*E_{pk}(110)^{330}\\&\quad*E_{pk}(0)^{96}*E_{pk}(31)^{570}.\\\mathcal{A}_3^{(\boldsymbol{q})}&=E_{pk}(10)^{510}*E_{pk}(95)^{45}*E_{pk}(110)^{249}\\&\quad*E_{pk}(0)^{63}*E_{pk}(31)^{420}.\\\mathcal{A}_4^{(\boldsymbol{q})}&=E_{pk}(10)^{900}*E_{pk}(95)^{87}*E_{pk}(110)^{480}\\&\quad*E_{pk}(0)^{600}*E_{pk}(31)^{186}.\\\mathcal{A}_5^{(\boldsymbol{q})}&=E_{pk}(10)^{330}*E_{pk}(95)^{840}*E_{pk}(110)^{108}\\&\quad*E_{pk}(0)^{690}*E_{pk}(31)^{390}.\end{aligned}$$

Lastly, QU decrypts $\boldsymbol{\mathcal{A}}^{(\boldsymbol{q})}$, and obtains

$$\begin{aligned}\boldsymbol{q}'&=D_{sk}\left(\boldsymbol{\mathcal{A}}^{(\boldsymbol{q})}\right)^T/100\\&=(191.25,1849.80,497.85,758.31,1070.70)^T\end{aligned}$$

where the divisor 100 is used to eliminate the scaling factors of DO and QU.

• kNNComp$(\mathbf{D}',\boldsymbol{q}',1)\mapsto\{\mathcal{I}_{\boldsymbol{q}}\}$: At this stage, CS uses $\mathbf{D}'=\{\boldsymbol{p}_1',\boldsymbol{p}_2'\}$ and $\boldsymbol{q}'$ to find out the 1-NN point of $\boldsymbol{q}$. Here, CS computes $\boldsymbol{p}_1'\boldsymbol{q}'=871.3506$ and $\boldsymbol{p}_2'\boldsymbol{q}'=1052.1975$. Because $\boldsymbol{p}_1'\boldsymbol{q}'<\boldsymbol{p}_2'\boldsymbol{q}'$, CS can infer that $\boldsymbol{p}_1$ is closer to $\boldsymbol{q}$ than $\boldsymbol{p}_2$, that is, the 1-NN point is $\boldsymbol{p}_1$, which completes the query.

In the toy example, the real distances are $\mathcal{D}(\boldsymbol{p}_1,\boldsymbol{q})=\sqrt{(10-9.5)^2+(2.7-3.1)^2}=0.64$, and $\mathcal{D}(\boldsymbol{p}_2,\boldsymbol{q})=\sqrt{(5.3-9.5)^2+(8.6-3.1)^2}=6.92$. Thus, $\boldsymbol{p}_1$ is indeed the closer one. It shows the query result of our scheme is correct.

4.3. Further improvement to support secure k-NN query over partial dimensions

In the real-world, some applications may require to find the k-NN points over partial dimensions, rather than all of them. For example, the database consists of age, sex, income, property, and several other attributes, but QU might hope to perform k-NN query on age and income only, due to the particular requirement of some application. In our system model, the database $\mathbf{D}$ has d dimensions. Assume the query $\boldsymbol{q}$ requires to find the k-NN points on only the dimensions indexed by $\mathcal{I}_{\boldsymbol{q}}$ where $\mathcal{I}_{\boldsymbol{q}}=\{I_1^{(\boldsymbol{q})},I_2^{(\boldsymbol{q})},\ldots,I_t^{(\boldsymbol{q})}\}\subseteq\{1,2,\ldots,d\}$ and $1\leqslant t\leqslant d$. That is, it requires to use the distance

$\mathcal{D}_{\mathcal{I}_q}(\boldsymbol{p}_i, \boldsymbol{q}) = \sqrt{\sum_{j\in\mathcal{I}_q}(p_{ij} - q_j)^2}$ while querying k-NN tuples. For any i and h, it is easy to see

$$\begin{aligned}\mathcal{D}^2_{\mathcal{I}_q}(\boldsymbol{p}_i, \boldsymbol{q}) - \mathcal{D}^2_{\mathcal{I}_q}(\boldsymbol{p}_h, \boldsymbol{q}) &= \sum_{j\in\mathcal{I}_q}(p_{ij} - q_j)^2 - \sum_{j\in\mathcal{I}_q}(p_{hj} - q_j)^2\\ &= \sum_{j\in\mathcal{I}_q}(p^2_{ij} - 2p_{ij}q_j) - \sum_{j\in\mathcal{I}_q}(p^2_{hj} - 2p_{hj}q_j)\\ &= \Big(\sum_{j\in\mathcal{I}_q} p^2_{ij} + \sum_{j\in\mathcal{I}_q}(-2p_{ij}q_j)\Big) - \Big(\sum_{j\in\mathcal{I}_q} p^2_{hj} + \sum_{j\in\mathcal{I}_q}(-2p_{hj}q_j)\Big).\end{aligned}$$

Then, apart from the scalar product $\sum_{j\in\mathcal{I}_q}(-2p_{ij}q_j)$, we need $\sum_{j\in\mathcal{I}_q} p^2_{ij}$ rather than $\|\boldsymbol{p}_i\|^2$, while computing the k-NN points on partial dimensions. Therefore, the scheme, in which the $(d+1)$-th dimension of $\dot{\boldsymbol{p}}_i$ is set as $\|\boldsymbol{p}_i\|^2$, cannot support the query over less than d dimensions.

Here, we will propose a further practical improvement such that the k-NN query can be securely implemented on arbitrary subset of the dimensions in DO's dataset, according to the dynamic requirement of each query.

While the k-NN query is required to perform on the dimensions indexed by $\mathcal{I}_q = \{I^{(q)}_1, I^{(q)}_2, \ldots, I^{(q)}_t\}$ (i.e., using the distance $\mathcal{D}_{\mathcal{I}_q}(\boldsymbol{p}_i, \boldsymbol{q}) = \sqrt{\sum_{j\in\mathcal{I}_q}(p_{ij} - q_j)^2}$), our scheme can complete the query through the following improvement.

First, we replace $(-2\boldsymbol{p}_i, \|\boldsymbol{p}_i\|^2)$ with a $(2d)$-dimensional vector $\boldsymbol{p}^*_i = (p^*_1, p^*_2, \ldots, p^*_{(2d)})$ while encrypting the point in dataset $\mathbf{D}$. Here each dimension of $\boldsymbol{p}^*_i$ satisfies

$$p^*_{ij} = \begin{cases} -2p_{ij}, & \text{if } 1 \leqslant j \leqslant d\\ p^2_{i(j-d)}, & \text{if } d < j \leqslant 2d.\end{cases} \tag{5}$$

Second, QU replaces $(\boldsymbol{q}, 1)$ with another $(2d)$-dimensional vector $\boldsymbol{q}^*$ in QueryEnc stage.

$$q^*_j = \begin{cases} q_j, & \text{if } 1 \leqslant j \leqslant d \text{ and } j \in \mathcal{I}_q\\ 1, & \text{if } d < j \leqslant 2d \text{ and } (j-d) \in \mathcal{I}_q\\ 0, & \text{otherwise.}\end{cases} \tag{6}$$

In our improvement above, the length of random vector S will be $2d$, and η will be $2d + c + \epsilon$.

We prove the *correctness* of the above improvement as follows.

In Section 4.1, Eqs. (3) and (4) have shown $\boldsymbol{p}'_i\boldsymbol{q}' - \boldsymbol{p}'_h\boldsymbol{q}' = \dot{\boldsymbol{p}}_i\dot{\boldsymbol{q}}^T - \dot{\boldsymbol{p}}_h\dot{\boldsymbol{q}}^T = \beta_{\boldsymbol{q}}\big((-2\boldsymbol{p}_i, \|\boldsymbol{p}_i\|^2)\cdot(\boldsymbol{q}, 1) - (-2\boldsymbol{p}_h, \|\boldsymbol{p}_h\|^2)\cdot(\boldsymbol{q}, 1)\big)$. After the replacements in our improvement, we thus have $\boldsymbol{p}'_i\boldsymbol{q}' - \boldsymbol{p}'_h\boldsymbol{q}' = \beta_{\boldsymbol{q}}\big(\boldsymbol{p}^*_i\cdot\boldsymbol{q}^* - \boldsymbol{p}^*_h\cdot\boldsymbol{q}^*\big)$.

According to the settings in Eqs. (5) and (6), we have $\boldsymbol{p}^*_i\cdot\boldsymbol{q}^* = \sum_{j\in\mathcal{I}_q} p^2_{ij} - 2\sum_{j\in\mathcal{I}_q} p_{ij}q_j$. Then,

$$\begin{aligned}\boldsymbol{p}^*_i\cdot\boldsymbol{q}^* - \boldsymbol{p}^*_h\cdot\boldsymbol{q}^* &= \sum_{j\in\mathcal{I}_q}(p^2_{ij} - 2p_{ij}q_j + q^2_j)\\ &\quad - \sum_{j\in\mathcal{I}_q}(p^2_{hj} - 2p_{hj}q_j + q^2_j)\\ &= \mathcal{D}^2_{\mathcal{I}_q}(\boldsymbol{p}_i, \boldsymbol{q}) - \mathcal{D}^2_{\mathcal{I}_q}(\boldsymbol{p}_h, \boldsymbol{q}).\end{aligned}$$

Further, $\boldsymbol{p}'_i\boldsymbol{q}' - \boldsymbol{p}'_h\boldsymbol{q}' = \beta_{\boldsymbol{q}}\big(\mathcal{D}^2_{\mathcal{I}_q}(\boldsymbol{p}_i, \boldsymbol{q}) - \mathcal{D}^2_{\mathcal{I}_q}(\boldsymbol{p}_h, \boldsymbol{q})\big)$. Because $\beta_{\boldsymbol{q}} > 0$, we can find the smaller distance on the required dimensions by comparing $\boldsymbol{p}'_i\boldsymbol{q}'$ and $\boldsymbol{p}'_h\boldsymbol{q}'$.

Therefore, in our improvement above, CS can compute the k-NN points by using the encrypted $\boldsymbol{p}'_i$ and $\boldsymbol{q}'$ as well. That is, our improved scheme can correctly support k-NN query over the partial dimensions indexed by $\mathcal{I}_q$. Additionally, QU can dynamically adjust $\mathcal{I}_q$ for each query.

5. Security analysis

5.1. Data privacy and query privacy

In our scheme, other than DO himself, only CS can access the outsourced database $\mathbf{D}'$, thus we only need to confirm the data privacy against the remote CS. For each dimension q_i of query point $\boldsymbol{q}$, DO only observes $E_{pk}(q_i)$, the ciphertext in Paillier cryptosystem [21]. Because the encryption in [21] is semantically secure, it can guarantee the query privacy against DO. For the query privacy against CS, we consider it together with data privacy against CS under level-2 attack as follows.

The work [35] has shown ASPE using a random invertible matrix to encrypt can resist level-2 attack, such as, signature linking attack [35], principle component analysis (PCA) [19], duplicate analysis [1], brute-force attack. Here, we do not discuss the details about the above level-2 attack approaches again. While computing the encrypted points $\boldsymbol{p}'_i$ and $\boldsymbol{q}'$, we first use tuple-sharing perturbation, dynamic virtual dimensions and a permutation to perturb plain points, and then attain our encrypted points through multiplying the perturbed tuples $\dot{\boldsymbol{p}}_i$ and $\dot{\boldsymbol{q}}$ by a random invertible matrix $\boldsymbol{M}$ (or $\boldsymbol{M}^{-1}$). It is obvious that our scheme is secure against each attack that ASPE [35] can resist at level-2, as ASPE [35] only uses a random invertible matrix to encrypt. Namely, the new scheme can resist all the aforecited level-2 attacking methods.

The apriori-knowledge independent component analysis (AK-ICA) [14,20] is also introduced to approximately reconstruct the original data perturbed by matrix transformation while some samples of original data are available to attackers. The attack method applies the traditional ICA to obtain the mixing matrices, and then derives an estimate of the transformation matrix based on the assumption that the known samples follow the same distribution with original dataset. Nevertheless, we introduce one-time random dimensions ($\boldsymbol{v}_i$ for database point $\boldsymbol{p}_i$, and $\boldsymbol{R}^{(\boldsymbol{q})}$ for query point $\boldsymbol{q}$) during generating the intermediate perturbed tuples $\dot{\boldsymbol{p}}_i$ and $\dot{\boldsymbol{q}}$. The one-time vectors $\boldsymbol{v}_i$ and $\boldsymbol{R}^{(\boldsymbol{q})}$ are independently set by DO each time and privately kept to DO, which can impede both ICA and estimating the transformation matrix in AK-ICA. Our scheme is therefore resilient to AK-ICA as well.

As a result, our scheme can preserve data privacy and query privacy against CS under level-2 attack.

5.2. Key confidentiality against QUs

In our scheme, QUs interact with DO during the QueryEnc stage only. This sub-section confirms that QUs cannot learn the key of DO in QueryEnc.

During encrypting query point $\boldsymbol{q}$, the QU can get only an encrypted vector $\mathcal{A}^{(\boldsymbol{q})}$. Then, for all $i \in [1, \eta]$, $q'_i = \beta_{\boldsymbol{q}}\boldsymbol{M}_{i*}\dot{\boldsymbol{q}}^T$ can be obtained by decrypting ith dimension of $\mathcal{A}^{(\boldsymbol{q})}$.

Firstly, we consider key confidentiality against QUs without permutation during computing $\dot{\boldsymbol{q}}$ in Eq. (2), and use the settings $\ddot{\boldsymbol{q}} = \boldsymbol{\pi}^{-1}(\dot{\boldsymbol{q}}) = \big(\boldsymbol{q}, 1, \boldsymbol{R}^{(\boldsymbol{q})}, \mathbf{0}_{(\epsilon)}\big)$ and $\ddot{\boldsymbol{q}}' = \beta_{\boldsymbol{q}}\boldsymbol{M}\ddot{\boldsymbol{q}}^T$. For ith dimension $\ddot{q}'_i$ of $\ddot{\boldsymbol{q}}'$, we have $\ddot{q}'_i = \beta_{\boldsymbol{q}}\boldsymbol{M}_{i*}\ddot{\boldsymbol{q}}^T$. Then, the QU can get the equation

$$\ddot{q}'_i = \beta_{\boldsymbol{q}}\Big(\sum_{j=1}^{d} M_{ij}q_j\Big) + \Big(\beta_{\boldsymbol{q}}M_{i,d+1} + \beta_{\boldsymbol{q}}\sum_{j=d+2}^{d+1+c} M_{ij}R^{(\boldsymbol{q})}_{j-d-1}\Big). \tag{7}$$

In Eq. (7), the values of $\{\beta_{\boldsymbol{q}}, M_{ij}, R^{(\boldsymbol{q})}_h : 1 \leqslant h \leqslant c,\ 1 \leqslant j \leqslant (d+1+c)\}$ all are unknown to the QU, and she only knows $\{\ddot{q}'_i, (q_1, q_2, \ldots, q_d)\}$. Let $X^{(i)}_{\boldsymbol{q}} = \sum_{j=1}^{d} M_{ij}q_j$, and $\zeta^{(i)}_{\boldsymbol{q}}$ be equal to $\big(\beta_{\boldsymbol{q}}M_{i,d+1} + \beta_{\boldsymbol{q}}\sum_{j=d+2}^{d+1+c} M_{ij}R^{(\boldsymbol{q})}_{j-d-1}\big)$. Based on the "encrypted query point" $\ddot{\boldsymbol{q}}'$, the following equation (for any $i \in [1, \eta]$) can be set up

$$\beta_{\boldsymbol{q}}X^{(i)}_{\boldsymbol{q}} + \zeta^{(i)}_{\boldsymbol{q}} = \ddot{q}'_i. \tag{8}$$

Due to the randomness of $\{M_{ij} : 1 \leqslant j \leqslant (d+1+c)\}$ and that $\{\beta_{\boldsymbol{q}}, \boldsymbol{R}^{(\boldsymbol{q})}\}$ are one-time random parameters to be independently re-selected for any new query point, $\zeta_{\boldsymbol{q}}^{(i)}$ is completely random to the QU even the user observes a big number of encrypted query points in the legal way or through collusion with other QUs. Thus, the QU only can obtain negligible information about $X_{\boldsymbol{q}}^{(i)}$, further, she cannot learn useful information about $\{M_{ij} : 1 \leqslant j \leqslant d\}$ hidden in $X_{\boldsymbol{q}}^{(i)}$.

Secondly, after applying the permutation, QUs will not learn the correspondence of the dimensions of $\ddot{\boldsymbol{q}}$ and $\boldsymbol{M}_{i*}$. Since the last ϵ dimensions of $\ddot{\boldsymbol{q}}$ are 0, the number of possible correspondence is $\eta!/\epsilon!$. For a real dataset 'Shuttle' from the UCI repository [31] which contains 9 dimensions, if DO sets the system parameters $c = 2$ and $\epsilon = 2$, then, $\eta = 14$ and $14!/2! = 43, 589, 145, 600$. Besides, QUs cannot efficiently distinguish each possible permutation, because $\beta_{\boldsymbol{q}}$, $\boldsymbol{R}^{(\boldsymbol{q})}$ and M_{ij} all are unknown. Consequently, the permutation can efficiently prevent QUs from setting up Eqs. (7) and (8), and it also enhances the security of our scheme.

In summary, QUs cannot learn the key of DO, and our scheme is capable to achieve key confidentiality against QUs.

5.3. Query controllability

Since DO shares his key with all QUs in many existing schemes, a QU can launch any query upon receiving the key. Even a QU is found to be suspect after she obtains the key, it is of much difficulty for DO to prohibit the QU, unless updating the key and re-encrypting the total database. The lax rules give the freedom to QUs, but bring about serious potential risks to DO and his outsourced dataset. Our scheme with query controllability aims at responding the above disadvantages and enabling DO to supervise each query on his database. Query controllability requires that any QU cannot launch a feasible query for a new given query point without approval of DO, i.e., the QU cannot deduce a feasible encrypted result for the given point if DO does not approve. Theorem 1 confirms our scheme can guarantee query controllability.

Theorem 1. *In our scheme, given a query point $\boldsymbol{q}_0$, any QU, who has not obtained a legal encrypted item of $\boldsymbol{q}_0$, cannot launch a feasible k-NN query for it without approval of DO.*

Proof. Since a QU, who has ever received several legal encrypted query points, is more possible to be capable of launching an uncontrolled query than a new QU, we discuss the possibility that an experienced QU breaks the controllability of our scheme.

Suppose the experienced QU has obtained the legal encrypted results of μ query points $\mathcal{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_\mu\}$ in which $\boldsymbol{q}_j \neq \boldsymbol{q}_0$ for all $j \in [1, \mu]$, and $\boldsymbol{q}'_j$ is the encrypted result of $\boldsymbol{q}_j$. If the QU can launch a feasible query for $\boldsymbol{q}_0$ without approval of DO, there exists a function $f(\cdot)$ such that an encrypted item $\boldsymbol{q}'_0$ can be computed by the fashion $\boldsymbol{q}'_0 = f(\boldsymbol{q}'_1, \boldsymbol{q}'_2, \ldots, \boldsymbol{q}'_\mu)$. We will show that $f(\cdot)$ does not exist while the QU has not legally launched $\boldsymbol{q}_0$, i.e., she does not learn the encrypted vector $\boldsymbol{q}'_0$ previously.

As the encryption of query points is substantially a linear transformation and QUs cannot learn the key of DO, $f(\cdot)$ must be a linear combination of encrypted values of points in $\mathcal{Q}$, i.e., $f(\boldsymbol{q}'_1, \boldsymbol{q}'_2, \ldots, \boldsymbol{q}'_\mu) = \sum_{j=1}^{\mu} x_j \boldsymbol{q}'_j$ where $x_j \in \mathbb{R}$. Thus, the QU can break query controllability if and only if she can determine a set $\{x_1, x_2, \ldots, x_\mu\}$ which enables the following conditions to hold

$$\beta_{\boldsymbol{q}_0} > 0, \quad \text{and} \quad \beta_{\boldsymbol{q}_0} \boldsymbol{M} \dot{\boldsymbol{q}}_0^T = \sum_{j=1}^{\mu} x_j \beta_{\boldsymbol{q}_j} \boldsymbol{M} \dot{\boldsymbol{q}}_j^T.$$

Since the matrix $\boldsymbol{M}$ is invertible, it can be reduced to the terms: $\beta_{\boldsymbol{q}_0} > 0$ and $\beta_{\boldsymbol{q}_0} \dot{\boldsymbol{q}}_0 = \sum_{j=1}^{\mu} x_j \beta_{\boldsymbol{q}_j} \dot{\boldsymbol{q}}_j$. Based on $\dot{\boldsymbol{q}} = \boldsymbol{\pi}\big(\boldsymbol{q}, 1, \boldsymbol{R}^{(\boldsymbol{q})}, \boldsymbol{0}_{(\epsilon)}\big)$ in Eq. (2), it requires the QU can fix a group of values for x_1 to x_μ such that

$$\begin{cases} \beta_{\boldsymbol{q}_0} > 0, \\ \beta_{\boldsymbol{q}_0} \boldsymbol{q}_0 = \sum_{j=1}^{\mu} x_j \beta_{\boldsymbol{q}_j} \boldsymbol{q}_j, \\ \beta_{\boldsymbol{q}_0} = \sum_{j=1}^{\mu} x_j \beta_{\boldsymbol{q}_j}, \\ \beta_{\boldsymbol{q}_0} \boldsymbol{R}^{(\boldsymbol{q}_0)} = \sum_{j=1}^{\mu} x_j \beta_{\boldsymbol{q}_j} \boldsymbol{R}^{(\boldsymbol{q}_j)}. \end{cases} \tag{9}$$

Since $\beta_{\boldsymbol{q}_0}$ must be positive, for $j \in [1, \mu]$, at least one $x_j \neq 0$ (otherwise $\beta_{\boldsymbol{q}_0} = \sum_{j=1}^{\mu} x_j \beta_{\boldsymbol{q}_j} \equiv 0$). Without loss of generality, let x_1 not equal to 0. While μ is big enough, the QU can figure out a set $\{\theta_j \in \mathbb{R} : 1 \leqslant j \leqslant \mu, \theta_1 \neq 0\}$ so that $\boldsymbol{q}_0 = \sum_{j=1}^{\mu} \theta_j \boldsymbol{q}_j$.

Then, $\beta_{\boldsymbol{q}_0} \boldsymbol{q}_0 = \sum_{j=1}^{\mu} \theta_j \beta_{\boldsymbol{q}_0} \boldsymbol{q}_j$. To determine x_j, the QU can set up $x_j \beta_{\boldsymbol{q}_j} = \theta_j \beta_{\boldsymbol{q}_0}$ according to Eq. (9).

Furthermore, $x_j \beta_{\boldsymbol{q}_j} = \theta_j \sum_{i=1}^{\mu} x_i \beta_{\boldsymbol{q}_i}$, for each $j \in [1, \mu]$.

Because each $\beta_{\boldsymbol{q}_j}$ is unknown to QUs and $x_1 \neq 0$, for $1 < j \leqslant \mu$, the QU can fix x_j if and only if $\theta_j = 0$, in which x_j will also be assigned to 0. Then, $\beta_{\boldsymbol{q}_0} = \sum_{j=1}^{\mu} x_j \beta_{\boldsymbol{q}_j} = x_1 \beta_{\boldsymbol{q}_1}$, and $x_1 \beta_{\boldsymbol{q}_1} = \theta_1 \sum_{i=1}^{\mu} x_i \beta_{\boldsymbol{q}_i} = \theta_1 x_1 \beta_{\boldsymbol{q}_1}$.

Therefore, the QU can determine a group of fixed values for x_1 to x_μ only when $\theta_2 = \theta_3 = \cdots = \theta_\mu = 0$ and $\theta_1 = 1$, and in this situation she can fix the values $x_2 = x_3 = \cdots = x_\mu = 0$ and assign any positive number to x_1. However, $\boldsymbol{q}_0 = \sum_{j=1}^{\mu} \theta_j \boldsymbol{q}_j = \boldsymbol{q}_1$, that is, the QU has launched a valid query for the point $\boldsymbol{q}_0$ in the legal manner. It leads to a contradiction, since the QU does not learn the encrypted vector $\boldsymbol{q}'_0$ previously.

To sum up, for a given query point $\boldsymbol{q}_0$, any QU, who has no legal encrypted result of it, cannot launch a feasible k-NN query for $\boldsymbol{q}_0$ without the approval of DO.

6. Performance evaluation

We analyze computation complexity and communication overheads of our scheme, then implement the new scheme, and compare it with the existing ASPE [35] and scheme in [42] which can accurately support secure k-NN query on encrypted cloud data in high efficiency (They can preserve data privacy and query privacy under level-2 attack, but achieve neither of query controllability and key confidentiality against QUs).

6.1. Cost analysis

Computation complexity. (I) During key generation, DO randomly selects an $\eta \times \eta$ invertible matrix $\boldsymbol{M}$, two vectors $\boldsymbol{S}$ of $(d+1)$ dimensions and $\boldsymbol{\tau}$ of c dimensions, and a permutation $\boldsymbol{\pi}$ of η numbers. Here $\eta = d + 1 + c + \epsilon$. Then, the computation complexity of DO in this stage is $O(\eta^2)$.

(II) The encryption of each database point includes $O(\eta)$ additions/multiplications, a permutation and a matrix multiplication. For all the m points in $\mathbf{D}$, this stage has $O(m\eta^2)$ computation overheads all of which are finished by DO.

(III) While computing an encrypted query point, QU generates the public key and secret key of Paillier cryptosystem [21], then uses her public key to encrypt d dimensions of her plain query point, and decrypts η numbers. In Paillier cryptosystem, key generation, one encryption and one decryption costs $O(1)$, $O(\log n)$ and $O(1)$ computation time, respectively. Therefore, the computation overheads of QU are $O(d \log n)$. In the stage, DO implements Algorithm 1 which has $O(\eta^2)$ multiplications and $O(c\eta)$ homomorphic encryptions, thus, his computation burden is $O(\eta^2 + c\eta \log n)$.

(IV) In the last stage, CS computes the k-NN by linearly scanning, which will take $O(m \log k)$ comparisons. For each comparison, it takes $O(\eta)$ multiplications/additions to compute the distance $\boldsymbol{p}_i'\boldsymbol{q}'$. The computation cost of CS for computing k-NN is $O(m\eta \log k)$.

Communication overheads. In our scheme, KeyGen and DBEnc are independently completed by DO, and no communication occurs in the two stages. We consider the communication overheads of the other stages. Assume that every index that CS returns to QU and each dimension of encrypted points both are b_0 bits.

In QueryEnc, $(d + \eta)$ encrypted numbers are transferred, thus, its bit-cost is $(d + \eta)b_0$.

During kNNComp, the communication overheads occur from submitting encrypted query point and returning the index set of k-NN. Since the encrypted points in our scheme are η-dimensional, the communication cost of uploading one encrypted query point to cloud is $b_0\eta$ bits. Together with b_0k bits for returning indexes to QU, this stage costs $(\eta + k)b_0$ bits in total.

6.2. Experiment results

We implement our scheme and two existing efficient algorithms: ASPE [35] and the scheme in [42], using C language. During executing our scheme, we utilize GMP library [29] and Paillier library [22] with key size of 1024 bits, and select the system parameters $c = 2$ and $\epsilon = 2$. Then, $\eta = d + 1 + c + \epsilon = d + 5$. All experiments are performed on Ubuntu with Intel Core i5 2.4 GHz CPU and 3.8 GB memory. For real numbers, the scaling factor is set as 10^6.

6.2.1. Key generation

Fig. 2 records the average key generation time over 1000 runs. The results show, in KeyGen stage, our scheme costs just a little more time than ASPE, both of which need less time compared with scheme in [42]. While the dimension d of plain points is as high as 100, the average time of key generation in our scheme is about 1.5×10^{-4} s. Since the scheme in [42] generates a large matrix of $(2d + 2)^2$ elements, its time consumption is about four times that of other two schemes, especially while d is as big as 100.

6.2.2. Database encryption

We evaluate the time to encrypt random databases consisting of 10^5 to (2×10^6) points the dimension of which ranges from 2 to 100. Fig. 3(a) shows the encryption time while $d = 10$ and the value of m is from 100K to 2000K (K $= 10^3$). For the dataset containing 500K points with dimension of 2–100, the encryption time is shown in Fig. 3(b). From the figures, we can see that ASPE is the most efficient one, because it computes the encrypted tuples through simply multiplying points by an invertible matrix. The database encryption time of our scheme is higher but very close to that of ASPE. While d is fixed, the encryption time scales linearly to the number of database points. During database encryption in the three methods, the most time-consuming step is always the matrix–vector multiplication. Thus, the scheme in [42] requires the most running time, because it uses a $(2d + 2) \times (2d + 2)$ matrix, compared with $(d + 1) \times (d + 1)$ in ASPE and $(d + 5) \times (d + 5)$ in our scheme. Fig. 3(b) shows that the difference between the scheme in [42] and other two approaches is more significant with increasing dimension d. Nevertheless, our scheme only costs almost the same encryption time with ASPE while d is close to 100. Concretely speaking, our scheme needs about 36.17 s and 89.45 s to encrypt two million 10-dimensional points and five hundred thousand 100-dimensional points, respectively. Since the response time requirement of database encryption is not stringent in real-world applications, our time consumption is still practical.

6.2.3. Query encryption

While encrypting a query point, the computation times of DO and QU in existing schemes [35,42] are less than 1 ms. Fig. 4 gives the computation time for encrypting one query point in our scheme, which is the average result over 1000 random query points. It shows that in our query encryption, for $d = 10$, the computation times of DO and QU are just 0.16 s and 0.07 s, respectively. Even when d reaches up to 100, QU's computation time is 0.6 s and DO's is about 1.23 s. The dimension is typically smaller than 100, and our execution time is relatively feasible.

6.2.4. k-NN computation

We set $k = 10$, and measure the k-NN computation time of CS on the encrypted datasets in the three schemes. For all datasets, the k-NN computation is completed by linearly scanning. Fig. 5 shows the implementation results which are averaged over 1000 random queries. As can be seen from Fig. 5, our scheme is a little slower than the most fast ASPE. The reason is that the dimension of encrypted points in our scheme is $(d+5)$, but that in ASPE is $(d+1)$. Scheme in [42] is the most time-consuming one, due to its longest $(2d + 2)$-dimensional encrypted points. With the growth of m and d, the computation time of all approaches almost linearly increases. For ($m = 2000$K, $d = 10$) and ($m = 500$K, $d = 100$), our scheme can always complete the k-NN computation within 15 s.

To summarize, although Paillier encryption system [21,22] is more expensive compared with linear transformation, we employ it only to protect query privacy against DO in the query encryption stage. Apart from query encryption, our implementation cost is very close to that of the most efficient ASPE. Due to utilizing Paillier cryptosystem, our query encryption costs more execution time, but the computation times of DO and QU in our scheme both are less than 1.25 s even when $d = 100$. If we just loose the query privacy against DO and plain query points are directly sent to DO in some scenario, each stage of our scheme will be almost as efficient as ASPE, nevertheless, we still can preserve the superior specialties: key confidentiality and query controllability.

7. Related work

So far, a big number of works have addressed various queries on encrypted dataset. We simply review part of them.

Song et al. [27] proposed a symmetric-key based searchable encryption scheme which enables user to ask keyword search on outsourced encrypted data. The work [3] presented a public-key based keyword search solution over encrypted text documents. Nevertheless, the query clients in both schemes hold the secret key to decrypt encrypted dataset. After them, several searchable encryption methods [32,10,37,2,17] were put forward to achieve higher efficiency and support more complex queries, but none of them consider secure k-NN computation.

Predicate encryption [4,26,18,33] can also be used to implement search functionalities on encrypted data. The scheme [33] can resist untrusted QUs, since DO privately keeps his master key. However, predicate encryption schemes [4,26,18,33] require expensive pairing computations, and they cannot efficiently support encrypted k-NN query in cloud.

The papers [7,8] presented the geometric perturbation method by which DO can perturb his privacy-sensitive data and export the perturbed dataset for data mining applications, including k-NN computation. Since the transformation scheme is not secure under level-2 attack [35,19], Wong et al. [35] propose ASPE scheme with higher security to support k-NN query on outsourced encrypted data. The work [39] presents another scheme providing interesting trade-offs between query cost and accuracy. In the scheme [16], encrypted dataset is stored at query clients and the clients bear search task for NN ($k = 1$) query. The setting

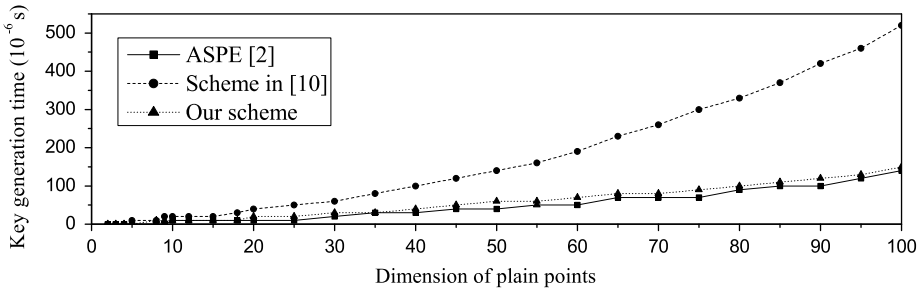
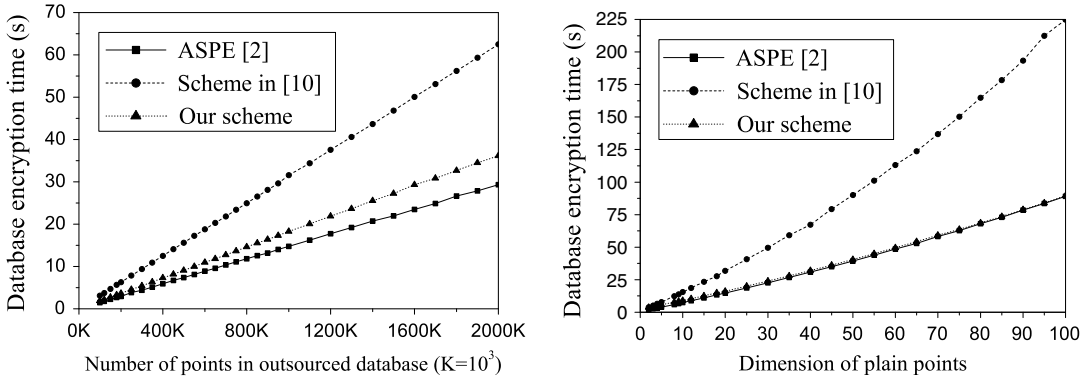


Fig. 2. Average time of key generation.



(a) $d = 10$. (b) $m = 5 \times 10^5$.

Fig. 3. Database encryption time.

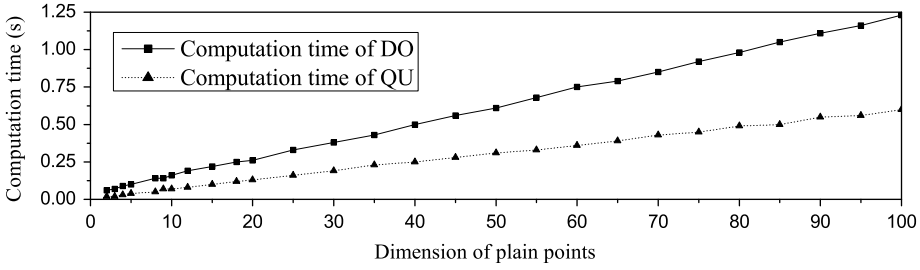
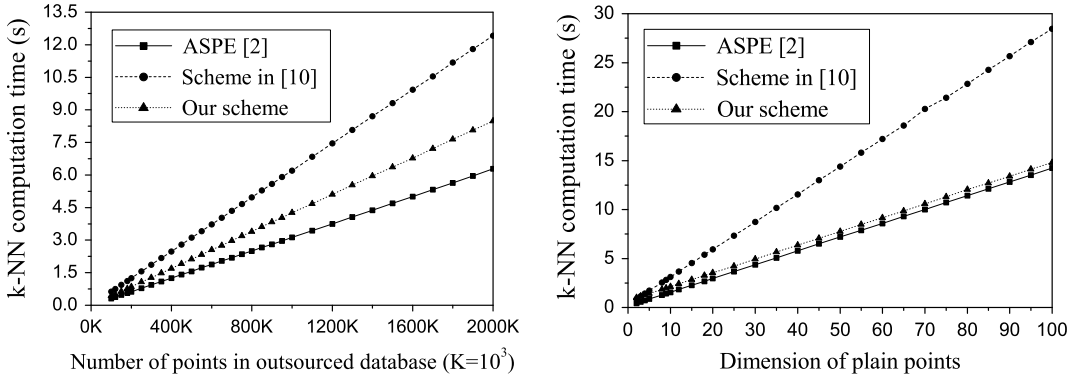


Fig. 4. Average computation time of query encryption in our scheme.



(a) $d = 10$. (b) $m = 5 \times 10^5$.

Fig. 5. Average k-NN computation time.

is different from ours, and the scheme in [16] introduces heavy computation burden to query clients. To achieve stronger IND-CPA (indistinguishability under chosen plaintext attack) security, [38] proposed an approximate NN scheme based on secure Voronoi diagram. RASP [36] also can be used to securely process k-NN query in an approximate manner. Elmehdwi et al. [25,11] lately presented a new accurate k-NN query approach on encrypted cloud data. Nevertheless, [25,11] require the existence of two non-colluding cloud service providers, and one of them learns the private key of DO, thus, it is still not realistic. Based on the mutable order-preserving encoding (mOPE) [23], Choi et al. [9] proposed a secure k-NN query approach in untrusted cloud environments.

Besides, Monomi [30] is put forward to implement any SQL query over encrypted data in cloud. However, nearly all the works [35,39,38,36,25,11,9,30] assume QUs are completely trusted and can access DO's key to encrypt/decrypt outsourced database. The wide distribution will inevitably increase the risk of key leakage and abuse, and it is of much difficulty to revoke the key distributed to QUs. Our previous work [42,43] considers the risk that QUs disclose their knowledge about the key of DO to attacker, but it still cannot achieve key confidentiality and query controllability. Yuan et al. declare their scheme in [40] can support secure NN query while resisting untrusted QUs and CS. Nevertheless, the attack approach proposed by Zhu et al. [41] shows that the encrypted dataset in [40] can be fast broken by untrusted QUs and CS. Additionally, DO in [40] can directly learn all the plain query vectors. Therefore, Yuan's scheme [40] can preserve neither data privacy nor query privacy.

8. Conclusion

In this paper, we addressed the privacy of DO and QUs during secure k-NN query on encrypted cloud data. Our proposed scheme can preserve key confidentiality and query controllability together with data privacy and query privacy. Theoretical analysis guaranteed the security and privacy properties. Through extensive experiments, we evaluated the overheads of our approach and compared it with existing works, which shows the efficiency and practicality of our new scheme.

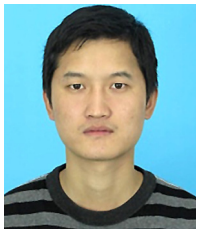
For the future work, we will devote to secure analysis scheme on encrypted cloud data with more realistic consideration, such as protecting the data access patterns from CS.

Acknowledgments

We wish to thank the anonymous reviewers for their perceptive and helpful suggestions on this paper. This work was supported in part by JSPS Grant-in-Aid (No. 24·02045), the National Natural Science Foundation of China (No. 61370224), the Fundamental Research Funds for the Central Universities (No. NZ2015108), the China Postdoctoral Science Foundation funded project (2015M571752), and the Jiangsu Planned Projects for Postdoctoral Research Funds (1402033C).

References

[1] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, Order preserving encryption for numeric data, in: Proc. of ACM SIGMOD, 2004, pp. 563–574.
[2] M. Bellare, A. Boldyreva, A. O'Neill, Deterministic and efficiently searchable encryption, in: CRYPTO, 2007, pp. 535–552.
[3] D. Boneh, G. Di Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in: Eurocrypt, 2004, pp. 506–522.
[4] D. Boneh, B. Waters, Conjunctive, subset, and range queries on encrypted data, in: Proc. of TCC, 2007, pp. 535–554.
[5] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, in: Proc. of IEEE INFOCOM, 2011, pp. 829–837.
[6] N. Cao, Z. Yang, C. Wang, K. Ren, W. Lou, Privacy-preserving query over encrypted graph-structured data in cloud computing, in: Proc. of 31st IEEE ICDCS, 2011, pp. 393–402.
[7] K. Chen, L. Liu, Privacy preserving data classification with rotation perturbation, in: Proc. of 5th IEEE ICDM, 2005.
[8] K. Chen, G. Sun, L. Liu, Towards attack-resilient geometric data perturbation, in: SIAM Data Mining Conference, 2007.
[9] S. Choi, G. Ghinita, H.-S. Lim, E. Bertino, Secure knn query processing in untrusted cloud environments, IEEE Trans. Knowl. Data Eng. 26 (11) (2014) 2818–2831.
[10] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved definitions and efficient constructions, in: Proc. of 13th ACM CCS, 2006, pp. 79–88.
[11] Y. Elmehdwi, B.K. Samanthula, W. Jiang, Secure k-nearest neighbor query over encrypted data in outsourced environments, in: IEEE 30th International Conference on Data Engineering, ICDE, 2014, pp. 664–675.
[12] B. Goethals, S. Laur, H. Lipmaa, T. Mielikainen, On private scalar product computation for privacy-preserving data mining, in: Proc. of 7th ICISC, in: LNCS, vol. 3506, 2004, pp. 104–120.
[13] O. Goldreich, Foundations of Cryptography: Volume II, Basic Applications, Cambridge University Press, Cambridge, 2004.
[14] S. Guo, X. Wu, Deriving private information from arbitrarily projected data, in: Proc. of the 11th PAKDD, in: LNAI, vol. 4426, Springer-Verlag, 2007, pp. 84–95.
[15] S. Han, W. Ng, L. Wan, V. Lee, Privacy-preserving gradient-descent methods, IEEE Trans. Knowl. Data Eng. 22 (6) (2010) 884–899.
[16] H. Hu, J. Xu, C. Ren, B. Choi, Processing private queries over untrusted data cloud through privacy homomorphism, in: Proc. of 27th IEEE ICDE, 2011, pp. 601–612.
[17] S. Kamara, C. Papamanthou, T. Roeder, Dynamic searchable symmetric encryption, in: Proc. of ACM CCS, 2012, pp. 965–976.
[18] J. Lai, X. Zhou, R.H. Deng, Y. Li, K. Chen, Expressive search on encrypted data, in: Proc. of 8th ASIACCS, 2013, pp. 243–252.
[19] K. Liu, C. Giannella, H. Kargupta, An attacker's view of distance preserving maps for privacy preserving data mining, in: Proc. of 10th PKDD, 2006, pp. 297–308.
[20] K. Liu, C. Giannella, H. Kargupta, Chapter 15: A survey of attack techniques on privacy-preserving data perturbation methods, in: Privacy-Preserving Data Mining, Springer, 2008, pp. 359–381.
[21] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: EUROCRYPT, in: LNCS, vol. 1592, 1999, pp. 223–238.
[22] Paillier library. http://acsc.cs.utexas.edu/libpaillier/.
[23] R.A. Popa, F.H. Li, N. Zeldovich, An ideal-security protocol for order-preserving encoding, in: Proc. of 34th IEEE Symposium on Security and Privacy, IEEE S&P, 2013, pp. 463–477.
[24] Y. Qi, M. Atallah, Efficient privacy-preserving k-nearest neighbor search, in: Proc. of 28th IEEE ICDCS, 2008, pp. 311–319.
[25] B.K. Samanthula, Y. Elmehdwi, W. Jiang, k-nearest neighbor classification over semantically secure encrypted relational data, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1261–1273.
[26] E. Shi, J. Bethencourt, T.-H. Chan, D. Song, A. Perrig, Multi-dimensional range query over encrypted data, in: IEEE Symposium on Security and Privacy, 2007, pp. 350–364.
[27] D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Proc. of IEEE Symposium on Security and Privacy, 2000, pp. 44–55.
[28] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, H. Li, Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, in: Proc. of 8th ASIACCS, ACM, 2013, pp. 71–82.
[29] The GNU multiple precision arithmetic library. http://gmplib.org/.
[30] S. Tu, M.F. Kaashoek, S. Madden, N. Zeldovich, Processing analytical queries over encrypted data, in: Proc. of the 39th VLDB, VLDB Endowment, 2013, pp. 289–300.
[31] UCI machine learning repository. http://archive.ics.uci.edu/ml/.
[32] C. Wang, N. Cao, K. Ren, W. Lou, Enabling secure and efficient ranked keyword search over outsourced cloud data, IEEE Trans. Parallel Distrib. Syst. 23 (8) (2012) 1467–1479.
[33] B. Wang, Y. Hou, M. Li, H. Wang, H. Li, Maple: Scalable multi-dimensional range search over encrypted cloud data with tree-based index, in: Proc. of 9th ASIACCS, ACM, 2014.
[34] C. Wang, Q. Wang, K. Ren, J. Wang, Harnessing the cloud for securely outsourcing large-scale systems of linear equations, IEEE Trans. Parallel Distrib. Syst. 24 (6) (2013) 1172–1181.
[35] W. Wong, D. Cheung, B. Kao, N. Mamoulis, Secure knn computation on encrypted databases, in: Proc. of 35th SIGMOD, 2009, pp. 139–152.
[36] H. Xu, S. Guo, K. Chen, Building confidential and efficient query services in the cloud with rasp data perturbation, IEEE Trans. Knowl. Data Eng.
[37] Z. Yang, S. Zhong, R.N. Wright, Privacy-preserving queries on encrypted data, in: ESORICS, 2006, pp. 479–495.
[38] B. Yao, F. Li, X. Xiao, Secure nearest neighbor revisited, in: Proc. of 29th IEEE ICDE, 2013, pp. 733–744.
[39] M.L. Yiu, I. Assent, C.S. Jensen, P. Kalnis, Outsourced similarity search on metric data assets, IEEE Trans. Knowl. Data Eng. 24 (2) (2012) 338–352.
[40] J. Yuan, S. Yu, Efficient privacy-preserving biometric identification in cloud computing, in: Proc. of IEEE INFOCOM, 2013, pp. 2652–2660.
[41] Y. Zhu, T. Takagi, R. Hu, Security analysis of collusion-resistant nearest neighbor query scheme on encrypted cloud data, IEICE Trans. Inf. Syst. E97-D (2) (2014) 326–330.
[42] Y. Zhu, R. Xu, T. Takagi, Secure k-nn computation on encrypted cloud data without sharing key with query users, in: Proc. of ASIACCS Workshop on Security in Cloud Computing, 2013, pp. 55–60.
[43] Y. Zhu, R. Xu, T. Takagi, Secure k-nn query on encrypted cloud database without key-sharing, Int. J. Electron. Secur. Digit. Forensics 5 (3–4) (2013) 201–217.



Youwen Zhu received his Ph.D. degree in Computer Science from University of Science and Technology of China in 2012. He is an Associate Professor at the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, and a JSPS Postdoctoral Fellow at the Institute of Mathematics for Industry, Kyushu University. His research interests include information security and privacy in cloud computing. His homepage is https://sites.google.com/site/zhuyouwen/. Email: zhuyouwen@gmail.com, zhuyw@nuaa.edu.cn.

Zhiqiu Huang is currently a Professor, the Dean of College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. He has published almost one hundred papers on major journals and conferences. His research interests include cloud security, privacy and software engineering. Email: zqhuang@nuaa.edu.cn.

Tsuyoshi Takagi received his Ph.D. degree with honors at the Department of Computer Science, Technische Universität Darmstadt in 2001. He is currently a Professor at the Institute of Mathematics for Industry in Kyushu University, Japan. He has published more than 100 papers in international journals and conferences. His research interests are mainly in the areas of information security and cryptography. For more details, please refer to his current personal webpage at Kyushu University http://imi.kyushu-u.ac.jp/~takagi/en/takagi.html.