Secure *k*-NN query on encrypted cloud database without key-sharing

Youwen Zhu*

Institute of Mathematics for Industry,
Kyushu University,
Fukuoka, 819-0395, Japan
E-mail: ywzhu@imi.kyushu-u.ac.jp
*Corresponding author

Rui Xu

Graduate School of Mathematics,
Kyushu University,
Fukuoka 819-0395, Japan
E-mail: r-xu@math.kyushu-u.ac.jp

Tsuyoshi Takagi

Institute of Mathematics for Industry,
Kyushu University,
Fuokuoka, 819-0395, Japan
E-mail: takagi@imi.kyushu-u.ac.jp

Abstract: In cloud computing, secure analysis on outsourced encrypted data is a significant topic. As a frequently used query for online applications, secure *k*-nearest neighbours (*k*-NN) computation on encrypted cloud data has received much attention, and several solutions for it have been put forward. However, most existing schemes assume the query users are fully trusted and all query users know the entire key which is used to encrypt and decrypt data owner's outsourced database. It is constitutionally not feasible in lots of real-world applications. In this paper, we propose a novel secure and practical scheme for preserving data privacy and supporting *k*-NN query on encrypted cloud data. In the new approach, only limited information about the key of data owner is disclosed to query users, and the data privacy can be protected even when query users leak their knowledge about the key to adversary. Theoretical analysis and experiment results confirm the security and practicality of our scheme.

Keywords: cloud computing; privacy; *k*-nearest neighbours; *k*-NN; query; key-sharing.

Reference to this paper should be made as follows: Zhu, Y., Xu, R. and Takagi, T. (2013) 'Secure *k*-NN query on encrypted cloud database without key-sharing', *Int. J. Electronic Security and Digital Forensics*, Vol. 5, Nos. 3/4, pp.201–217.

Biographical notes: Youwen Zhu received his PhD in Computer Sciences from University of Science and Technology of China. He is a JSPS Postdoctoral Fellow at the Institute of Mathematics for Industry in Kyushu University. His research interests include information security and privacy in cloud computing.

Rui Xu is a PhD candidate at the Graduate School of Mathematics in Kyushu University, Japan. His research interest is applied cryptography.

Tsuyoshi Takagi received his PhD degree with honours at the Department of Computer Science, Technische Universität Darmstadt. He is currently a Professor at the Institute of Mathematics for Industry in Kyushu University, Japan. His research interests are mainly in the areas of information security and cryptography.

This paper is a revised and expanded version of a paper entitled 'Secure k-NN computation on encrypted cloud data without sharing key with query users' presented at ASIACCS International Workshop on Security in Cloud Computing (CloudComputing'13), Hangzhou, 8 May 2013.

1 Introduction

In the last few years, cloud computing is becoming a more and more prevalent computing paradigm. At the same time, much attention has been paid to consider the special security and privacy problems in cloud computing. On the one hand, as cloud computing can provide convenient pay-as-you-go storage space, huge data are being increasingly centralised into the cloud to enjoy the coherence and economies of scale. However, it also arouses the security and privacy concerns, since the direct control will be transferred to cloud service provider while data owner outsources his dataset to a remote cloud server. Thus, data owner has to encrypt the sensitive information of his outsourced data, such as income level, health records, personal photos (Kamara and Lauter, 2010) before the dataset is uploaded to the cloud server such that his privacy is not breached. On the other hand, data owner may plan to make use of the strong computation ability of cloud service provider to process or query the database stored in the cloud server to obtain beneficial knowledge. However, the goal of most existing traditional encryption schemes is to protect the hidden plaintext, and their ciphertext cannot be as smoothly processed/analysed as the plain dataset. Therefore, a big number of efficient secure schemes have been proposed (Wong et al., 2009; Singh et al., 2010; Hu et al., 2011; Hore et al., 2012; Wang et al., 2010, 2012a, 2012b) to support the execution of application on encrypted data in the new promising cloud computing paradigm.

As the fundamental data mining query operation, the goal of *k*-nearest neighbours (*k*-NN) computation (Qi and Atallah, 2008; Shaneck et al., 2006) is to search *k* nearest points of a given query point according to some distance metric measures, such as Minkowski distance, Euclidean distance, and edit distance. To securely process *k*-NN query on outsourced encrypted data, Wong et al. (2009) proposed an asymmetric scalar-product-preserving encryption (ASPE) scheme. ASPE can achieve better security through using an invertible matrix, instead of orthogonal matrix in Chen and Liu (2005), and Oliveira and Zaiane (2003), to transform the database point. Heretofore, Wong's

ASPE scheme has been used as a black-box in many problems (Cao et al., 2011a, 2011b; Li et al., 2012). However, in ASPE, data owner will share the key for encryption and decryption with all query users, thus ASPE only supports fully trusted query users to conduct the *k*-NN query on encrypted cloud data. It will bring about serval problems. Firstly, since each query user has the access to the entire key, the adversary can obtain the key if he successfully decomposes one untrustworthy query user, i.e., the wide distribution will seriously increase the risk of key leakage. Secondly, in numerous concrete situations, data owner only has scant trust on each query user. For instance, some online social networking service site can take advantages of cloud service, and outsource the database of its members to cloud server for reducing the expense and enjoying other benefits of cloud. Simultaneously, its members may want to search the *k*-NN members with some similar social proximity. If using ASPE scheme, the members will encrypt the attributes with the same key as the one that the site encrypts the outsourced database. However, it is not realistic, since the site cannot share the key with the members otherwise its business confidentiality and members' privacy may be violated. Thirdly, data owner cannot control the queries of the users who has received the key, and it is of much difficulty for data owner to revoke the key of a query user. At last but not the least, a malicious cloud server may employ some bot-net user to steal the key in this system. Generally speaking, the existing query schemes which allow query users to have the access to the key of data owner are still far from being feasible in most real-world situations.

In this paper, considering a part of the above problems, we study the secure *k*-NN query computation on encrypted cloud data in the application situation that:

1 data owner encrypts his private database and outsources the encrypted dataset to cloud server for enjoying the storage and computation ability of the cloud

2 cloud server is honest-but-curious, that is, he will exactly follow the demand of data owner and query user to perform the computation but he also tries to infer as much private original value of encrypted outsourced dataset as possible by analysing the information from data owner and query user

3 each query user would like to find out the *k*-NN of her private query point without disclosing any privacy to data owner and cloud server, and before being submitted to cloud server for *k*-NN computation, her query point will be encrypted by using the key associated with the one of data owner such that cloud server can smoothly compute the *k*-NN while learning nothing about the original query point, but data owner cannot reveal his key to query user because query user is not trustworthy enough and she may reveal her knowledge about the key to the adversary (the cloud server) or it will seriously violate the business secret and privacy.

For the problems, a simple solution is that data owner and query user collaboratively perform a secure two-party computation protocol (Yao, 1982; Goldreich, 2004), which takes the key of data owner and query user's private point as input, such that query user receives the encrypted query points and data owner learns nothing. However, most of secure computation protocols are based on garbled-circuit protocol (Goldreich, 2004; Yao, 1986; Goldreich et al., 1987), secret sharing (Pedersen, 1991; Damgård et al., 2006) and homomorphic encryption system (Paillier, 1999) which are highly inefficient and thus not feasible for practical use. We consider the aforementioned problems and propose

a new secure *k*-NN query scheme. Similar to the existing schemes (Wong et al., 2009; Chen and Liu, 2005; Oliveira and Zaiane, 2003), the novel scheme is also a linear transformation and it can efficiently and effectively meet the requirements of our above application. A significant advantage of our scheme is that data owner will only release partial information of his key to query users, instead of the complete disclosure in existing schemes (Wong et al., 2009; Chen and Liu, 2005; Oliveira and Zaiane, 2003), and our scheme can still preserve data privacy even when some untrustworthy query users leak their knowledge about the key to adversary.

The rest of this paper is organised as follows. Section 2 discusses the related work. In Section 3, we introduce the system model, design goals and adversary models. In Section 4, we propose the new secure cloud storage and *k*-NN query scheme. Section 5 analyses the correctness, cost and security of our approach. We show and discuss the results of experiment evaluations in Section 6. At last, Section 7 concludes this paper.

2 Related work

During the past decade, some schemes for *k*-NN computation on encrypted/perturbed data have been proposed in the area of privacy-preserving data mining (Oliveira and Zaiane, 2003; Aggarwal and Philip, 2008; Chen et al., 2007). Their transformation schemes mainly consist of noise addition and random rotation perturbation, which are distance-preserved. It has been shown the distance-preserved transformation scheme is not secure when only a few points are disclosed (Wong et al., 2009; Liu et al., 2006, 2008). To deal with the problem and achieve better security, Wong et al. (2009) proposed another secure *k*-NN outsourced query scheme on encrypted data, ASPE. This scheme only preserves the distance between query point and the tuple in outsourced database, but it is impossible to recover the original distance between tuples in the encrypted outsourced dataset. The distance-unrecoverable property enables ASPE to resist against a strong adversary who not only observes the encrypted dataset but also knows some plain tuples in the original database.

As the nice properties in security and efficiency, ASPE has been used as the basic building of many secure query solutions (Cao et al., 2011a, 2011b; Li et al., 2012). Nevertheless, nearly all of them directly apply ASPE as a sub-step, and same with ASPE, they also require that data owner share the full key with each query user. The work (Wong et al., 2009) also enhances ASPE, by utilising random asymmetric splitting and adding artificial dimensions, to resist a stronger attacker that also learns a set of original tuples and their corresponding encrypted items (similar to chosen-plaintext attack, i.e., CPA attack). Nevertheless, Yao et al. (2013) show the enhanced ASPE cannot attain the security against CPA attack, and no accurate *k*-NN query scheme on encrypted cloud data can securely resist CPA attack. Then, some approximate *k*-NN query schemes on encrypted data are lately proposed in Yao et al. (2013) and Xu et al. (2012). However, they still consider only the fully trusted authorised query users, where data owner and query users share the same key and are assumed to entirely trust each other.

In this paper, we consider the security risk of releasing the full key to query users, and propose an improved scheme to preserve data privacy and support *k*-NN query on encrypted cloud database. In our scheme, query users can efficiently conduct the *k*-NN computation on cloud server, but no query user can obtain the full key of data owner.

Additionally, the data privacy still can be protected even query users reveal their knowledge to adversary.

Some other works (Kantarcioglu and Clifton, 2004; Xiong et al., 2007; Qi and Atallah, 2008; Amirbekyan Amirbekyan, 2007; Hashem et al., 2010; Papadopoulos et al., 2010; Ghinita et al., 2011) also focus on secure/privacy-preserving *k*-NN query to deal with the situations that query points and searching database are sensitive and not available at a single site. Nevertheless, their settings are generally different from our model. The schemes (Kantarcioglu and Clifton, 2004; Xiong et al., 2007; Qi and Atallah, 2008; Amirbekyan and Estivill-Castro, 2007) consider the model that database is vertically or horizontally distributed in two or more independent parties who want to compute the *k*-NN set of a public or confidential point without disclosing private data to each other. Their solutions are based on secure multiparty computation (Yao, 1982; Goethals et al., 2004; Goldreich, 2004), and the participants have equal/similar contribution in privacy-preserving computation, which is different from ours. Another type of work (Hashem et al., 2010; Papadopoulos et al., 2010; Ghinita et al., 2011) is recently prosperous to handle location privacy in location-based services (LBS), where LBS possesses a database of points of interest (POIs), and a client would like to find out *k* nearest POIs on the database without revealing his location. The solutions are usually to integrate private information retrieval (PIR) with LBS. Since the LBS can access the plain POIs database, their approaches cannot be used to protect the security of outsourced dataset.

3 Problem statement and design goals

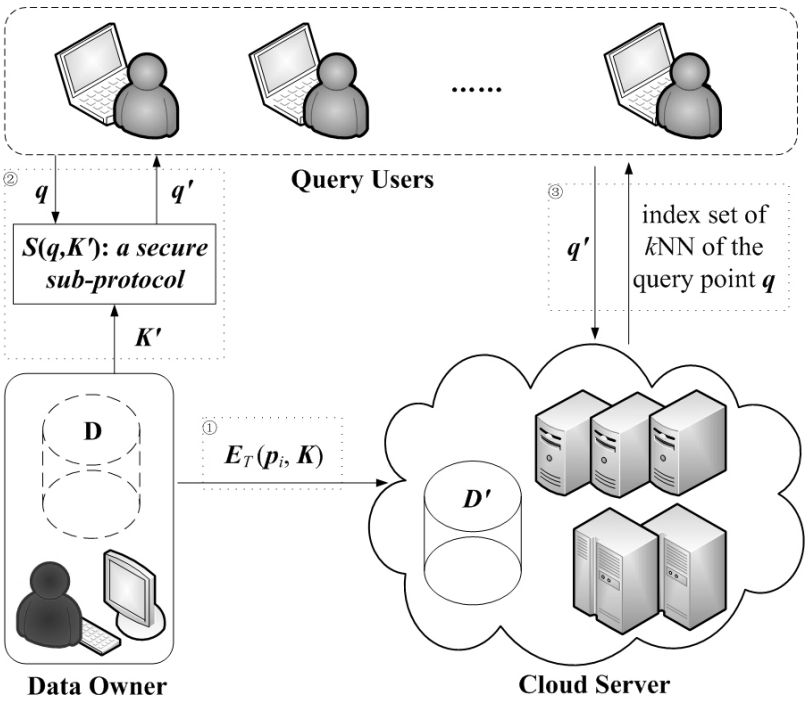
In this section, we introduce our system model, design goals and the framework of our new solution.

3.1 System model

This paper focuses on the security and privacy problems of cloud data storage and query service, which involves three types of entities: a data owner, a cloud server and some query users. They are shown in Figure 1. The cloud server owns huge storage and computing resources. The data owner possesses a large private database **D** consisting of *m d*-dimensional points: $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_m \in \mathbb{R}^d$. In practice, we have $d << m$, d is less than 100, but m would be 100,000 or larger. For enjoying the storage resources and computational ability of the cloud service provider, the database **D** of data owner will be outsourced to the cloud server in the encrypted form $\mathbf{D}' = \{\boldsymbol{E}_T(\boldsymbol{p}_1), \boldsymbol{E}_T(\boldsymbol{p}_2), \cdots, \boldsymbol{E}_T(\boldsymbol{p}_m)\}$. Each query user holds a private query point $\boldsymbol{q} \in \mathbb{R}^d$ with the same formation as the point in **D**. The query user would like to know the index set of the *k*-NN in **D** of his query point, according to the squared Euclidean distance $D(\boldsymbol{p}_i, \boldsymbol{q}) = \sum_{j=1}^{d} (p_{ij} - q_j)^2$. For the privacy of query user, q is also encrypted before it is sent to data owner and cloud server. During the query processing of our scheme, data owner and the corresponding query user will compute the encrypted query point $\boldsymbol{q}'$ in a collaborative manner such that only the query user obtains $\boldsymbol{q}'$. At last, query user submits the encrypted vector $\boldsymbol{q}'$ to cloud

server, then, the server executes an encrypted query on the encrypted database and returns the index set of *k*-NN of ***q*** to the corresponding query user. In this paper, a tuple is also called as a point, and we use the words 'tuple' and 'point' interchangeably.

Figure 1 Architecture of *k*-NN queries on encrypted cloud data



3.2 Design goals and adversary models

In our model, we consider the privacy of data owner and query user, but not the privacy of cloud server, as it just rents out the huge storage and computation power but inputs no private data. Additionally, the encryption and query computation should be efficient, and most computation should be done at the cloud server.

- *Data privacy*. For data privacy, the plain dataset **D** should be private to data owner throughout the outsourcing and *k*-NN computation, and it cannot be disclosed to cloud server and any query user. To prevent cloud server from learning the private database of data owner, only encrypted data will be stored in cloud server. Apart from the encrypted dataset **D**', the cloud server (potential attacker) is supposed to know-some original tuple in **D**, which is the same as known-sample attack in (Liu et al., 2006, 2008) or level-2 attack in Wong et al. (2009). We also consider the data privacy while cloud server obtains query users' knowledge about the key to encrypt and decrypt the outsourced database that data owner releases to query users during *k*-NN query. In this paper, we do not cope with the malicious query users who disclose all their information (including private query points) to attacker, which we leave as the future work. The adversary models of this work can be generalised as the following three levels.

1 The adversary only knows the encrypted dataset, including **D**' and encrypted query points. This level-1 adversary model is corresponding to the only-ciphertext attack.

2 The adversary learns some original points (but does not the corresponding encrypted items) except the encrypted dataset. The level-2 adversary model is the same as the level-2 attack in Wong et al. (2009).

3 In addition to the encrypted dataset, the adversary also obtains the knowledge about key that data owner reveals to query users. This is the novel level-3 adversary model in this paper.

- *Query privacy*. The query privacy requires that each query point is privately kept to the corresponding query user during the outsourced *k*-NN computation. We do not consider the security of query points against the collusion attack of data owner and cloud server, as the collusion of data owner and cloud server can inherently infer the query point with high accuracy from its *k*-NN, which is the output of the query. Since query user will send the perturbed query point to data owner and cloud server, we will consider the query privacy against data owner and cloud server, respectively.
- *Efficiency*. Additionally, our scheme will preserve data privacy and query privacy in an efficient and practical manner. That is, the above goals of security and privacy should be achieved with practically low computation cost and communication overheads. Concretely speaking, data owner should can efficiently encrypt each private tuple in **D**, each query user's computation and communication overheads should be practically low, and computing *k*-NN on the encrypted database should not notably increase the cost of cloud server comparing to using plain database.

4 Secure *k*-NN query scheme without key-sharing

In this section, we propose our new solution, and the analysis of it in correctness, cost and security.

4.1 Notations

- **D** – the private database of data owner, consisting of m points $\mathbf{D} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_m\}$
- $\mathbf{D}'$ – the encrypted database of **D**, being stored in cloud server, consisting of m points $\mathbf{D}' = \{\boldsymbol{p}'_1, \boldsymbol{p}'_2, \cdots, \boldsymbol{p}'_m\}$ in which $\boldsymbol{p}'_i = E_T(\boldsymbol{p}_i)$ is the encrypted form of $\boldsymbol{p}_i$
- $\boldsymbol{q}$ – a query point
- $\boldsymbol{q}'$ – the encrypted result of q
- $D(\boldsymbol{p}_i, \boldsymbol{q})$ – the distance of $\boldsymbol{p}_i$ and $\boldsymbol{q}$, and it equals to $\sum_{j=1}^{d} \left(p_{ij} - q_j \right)^2$
- a $L \times C$ matrix M – a matrix which has L lines and C columns, and it can also be denoted as matrix $M_{L\times C}$ in this paper
- $\boldsymbol{M}_{j*}$ – the j^{th} row of matrix $\boldsymbol{M}$

- $\mathcal{I}_q$ – the index set of the k-NN of the query point $\boldsymbol{q}$
- $[X]$ – the set $\{1, 2, \cdots, X\}$ for any positive integer X
- $\mathbb{R}$ – the real number field.

4.2 Framework

Our solution in this paper consists of four stages: KeyGen, TupleEnc, QueryEnc and OutKNN, the general functions of which are illustrated as follows.

- $\mathsf{keyGen}(1^{\varkappa}) \mapsto \{Key\}$: This stage will be completed by only the data owner. It takes a system security parameter $\varkappa$ as input, and returns a random secret Key which is the key used to later encrypt each tuple of private database $\mathbf{D}$ and the private query point $\boldsymbol{q}$.
- $\mathsf{TupleEnc}(\mathbf{D}, Key) \mapsto \{\mathbf{D}'\}$: In this stage, data owner will locally encrypt each tuple in $\mathbf{D}$. Each point is encrypted as a single unit, and $\boldsymbol{p}_i'$ is the encrypted item of $\boldsymbol{p}_i$. The output encrypted database is denoted as $\mathbf{D}' = \{\boldsymbol{p}_i' \mid \boldsymbol{p}_i \in \mathbf{D}\}$. The encrypted tuples will be sent to cloud server for storage and k-NN query.
- $\mathsf{QueryEnc}(\boldsymbol{q}, Key, 1^{\varkappa_q}) \mapsto \{\boldsymbol{q}'\}$: Taking the query user's private point $\boldsymbol{q}$, a security parameter $\boldsymbol{q}$ and the secret Key of data owner as inputs, this stage enable query user and data owner to cooperatively encrypt the query point $\boldsymbol{q}$ such that only query user obtains $\boldsymbol{q}'$ which is the encrypted result of $\boldsymbol{q}$. Besides, only partial information of Key will be disclosed to query users during the protocol.
- $\mathsf{OutKNN}(\mathbf{D}', \boldsymbol{q}', k) \mapsto \{\mathcal{I}_q\}$: Upon receiving an encrypted query $\boldsymbol{q}'$, cloud server computes the k-NN of the input query point through linearly scanning $\mathbf{D}'$, and returns $\mathcal{I}_q$ (the index set of the k-NN) to the corresponding query user.

4.3 Our scheme

We introduce our scheme by describing the explicit implementations of each stage, respectively.

- $\mathsf{keyGen}(1^{\varkappa}) \mapsto \{Key\}$: Data owner runs $\mathsf{keyGen}(1^{\varkappa})$ to randomly generate an invertible matrix $\boldsymbol{M} \in \mathbb{R}^{(2d+2)\times(2d+2)}$, a positive real number $\alpha \in \mathbb{R}^+$, two $(d+1)$-dimensional vectors $r = (r_1, r_2, \cdots, r_{d+1})$ and $s = (s_1, s_2, \cdots, s_{d+1}) \in \mathbb{R}^{d+1}$.

  Then, data owner sets $Key = \{\alpha, r, s, \boldsymbol{M}, \boldsymbol{M}^{-1}\}$, and keeps the Key in private.
- $\mathsf{TupleEnc}(\mathbf{D}, Key) \mapsto \{\mathbf{D}'\}$: The encryption of tuples includes two steps.

  1 For each $i \in [m]$, data owner computes the $(2d+2)$-dimensional vector

$$\dot{\boldsymbol{p}}_i = \left(r_1 - 2\alpha p_{i1}, s_1, r_2 - 2\alpha p_{i2}, s_2, \cdots, r_d - 2\alpha p_{id}, s_d, r_{d+1} + \alpha \sum_{j=1}^{d} p_{ij}^2, s_{d+1} \right).$$

2 Then, the encrypted item of $\boldsymbol{p}_i$ is obtained as follows:

$$\boldsymbol{p}_i' = \dot{\boldsymbol{p}}_i \boldsymbol{M}^{-1}.$$

We use $\mathbf{D}'$ to denote the encrypted database, that is, $\mathbf{D}' = \{\boldsymbol{p}_1', \boldsymbol{p}_2', \cdots, \boldsymbol{p}_m'\}$. After completing the encryption of private database $\mathbf{D}$, data owner uploads $\mathbf{D}'$ to cloud server for storage and on-line applications.

- $\mathsf{QueryEnc}(\boldsymbol{q}, Key, 1^{\lambda_{\boldsymbol{q}}}) \mapsto \{\boldsymbol{q}'\}$: The query encryption stage in our scheme consists of the following three steps.

  1 Query user generates a d-dimensional random vector $\boldsymbol{t} = (t_1, t_2, \cdots, t_d) \in \mathbb{R}^d$, and locally computes $\dot{q}_l = q_l + t_l$ for each $l \in [d]$. Then, he sends $[\dot{q}_1, \dot{q}_2, \cdots, \dot{q}_d]$ to data owner.

  2 Data owner randomly selects a real number $u \in \mathbb{R}$ and a positive real number $\beta_{\boldsymbol{q}} \in \mathbb{R}^+$, and computes a $(2d + 2)$-dimensional vector $\mathbf{X}$ and a $(2d + 2) \times (d)$ matrix $\boldsymbol{Y}$ by the following way,

$$X_j = \beta_{\boldsymbol{q}} \left(M_{j,2d+1} + uM_{j,2d+2} + \sum_{l=1}^{d} \dot{q}_l \left(2M_{j,2l-1} - M_{j,2l} \right) \right),$$
$$Y_{jl} = \beta_{\boldsymbol{q}} \left(M_{j,2l} - M_{j,2l-1} \right) \qquad j \in [2d+2], l \in [d].$$

  Here, X_i is the i^{th} element of $\boldsymbol{X}$, Y_{il} is the value in row i, column l of $\boldsymbol{Y}$, and M_{il} is the value in row i, column l of $\boldsymbol{M}$. Then, data owner sends $\boldsymbol{X}$ and $\boldsymbol{Y}$ to query user.

  3 At last, query user obtains the encrypted query point $\boldsymbol{q}' = (q_1', q_2', \cdots, q_{2d+2}')$ in which

$$q_j' = X_j + \sum_{l=1}^{d} Y_{jl} \left(q_l + 2t_l \right) \qquad j \in [2d+2].$$

- $\mathsf{OutKNN}(\mathbf{D}', \boldsymbol{q}', k) \mapsto \{\mathcal{I}_{\boldsymbol{q}}\}$: Query user uploads $\boldsymbol{q}'$ to cloud server, then cloud server computes the index set of k-NN in $\mathbf{D}'$ of the encrypted query point $\boldsymbol{q}'$ according to the distance $\boldsymbol{p}_i'(\boldsymbol{q}')^T$, and sends the index set $\mathcal{I}_{\boldsymbol{q}}$ of the k-NN in $\mathbf{D}'$ to the corresponding query user.

5 Theoretical analysis

5.1 Correctness analysis

We guarantee the correctness of our scheme through the following two theorems.

Theorem 1: In our scheme, the following equation (1) holds.

$$\begin{cases} \boldsymbol{p}_i'(\boldsymbol{q}')^T > \boldsymbol{p}_h'(\boldsymbol{q}')^T \ iff. D(\boldsymbol{p}_i, \boldsymbol{q}) > D(\boldsymbol{p}_h, \boldsymbol{q}), \\ \boldsymbol{p}_i'(\boldsymbol{q}')^T = \boldsymbol{p}_h'(\boldsymbol{q}')^T \ iff. D(\boldsymbol{p}_i, \boldsymbol{q}) = D(\boldsymbol{p}_h, \boldsymbol{q}), \\ \boldsymbol{p}_i'(\boldsymbol{q}')^T < \boldsymbol{p}_h'(\boldsymbol{q}')^T \ iff. D(\boldsymbol{p}_i, \boldsymbol{q}) < D(\boldsymbol{p}_h, \boldsymbol{q}). \end{cases} \tag{1}$$

Here, iff. denotes 'if and only if'.

Proof: In QueryEnc stage, for the j^{th} element of $\boldsymbol{q}'$, there is

$$\begin{aligned} q_j' &= X_j + \sum_{l=1}^{d} Y_{jl}(q_l + 2t_l) \\ &= \beta_{\boldsymbol{q}}\left(M_{j,2d+1} + uM_{j,2d+2} + \sum_{l=1}^{d} \dot{q}_l(2M_{j,2l-1} - M_{j,2l})\right) \\ &\quad + \sum_{l=1}^{d} \beta_{\boldsymbol{q}}(M_{j,2l} - M_{j,2l-1})(q_l + 2t_l) \\ &= \beta_{\boldsymbol{q}}(M_{j,2d+1} + uM_{j,2d+2}) \\ &\quad + \beta_{\boldsymbol{q}} \sum_{l=1}^{d} \left(\dot{q}_l(2M_{j,2l-1} - M_{j,2l}) + (M_{j,2l} - M_{j,2l-1})(q_l + 2t_l)\right). \end{aligned}$$

As $\dot{q}_l = q_l + t_l$, then, we have

$$\begin{aligned} &\dot{q}_l(2M_{j,2l-1} - M_{j,2l}) + (M_{j,2l} - M_{j,2l-1})(q_l + 2t_l) \\ &= 2q_lM_{j,2l-1} - q_lM_{j,2l} + 2t_lM_{j,2l-1} - t_lM_{j,2l} + q_lM_{j,2l} \\ &- q_lM_{j,2l-1} + 2t_lM_{j,2l} - 2t_lM_{j,2l-1} \\ &= q_lM_{j,2l-1} + t_lM_{j,2l}. \end{aligned}$$

Thus,

$$q_j' = \beta_{\boldsymbol{q}} \sum_{l=1}^{d} (q_lM_{j,2l-1} + t_lM_{j,2l}) + \beta_{\boldsymbol{q}}(M_{j,2d+1} + uM_{j,2d+2}).$$

Let $\boldsymbol{M}_{j*}$ be the j^{th} row of matrix $\boldsymbol{M}$ and $\ddot{\boldsymbol{q}}$ denote the $(2d + 2)$-dimensional vector $(q_1, t_1, q_2, t_2, \cdots, q_d, t_d, 1, u)$, then $q_j' = \beta_{\boldsymbol{q}}\boldsymbol{M}_{j*}\ddot{\boldsymbol{q}}^T$ and $\boldsymbol{q}' = \beta_{\boldsymbol{q}}\boldsymbol{M}\ddot{\boldsymbol{q}}^T$.

Therefore, $\boldsymbol{p}_i'(\boldsymbol{q}')^T = \dot{\boldsymbol{p}}_i\boldsymbol{M}^{-1}\beta\boldsymbol{M}\ddot{\boldsymbol{q}}^T = \beta\dot{\boldsymbol{p}}_i\ddot{\boldsymbol{q}}^T$. Further, we have

$$\begin{aligned} \boldsymbol{p}_i'(\boldsymbol{q}')^T &= \beta \sum_{j=1}^{d} (r_j - 2\alpha p_{ij})q_j + \beta \sum_{j=1}^{d} s_jt_j + r_{d+1}\beta + \beta \sum_{j=1}^{d} \alpha p_{ij}^2 + s_{d+1}u \\ &= \alpha\beta \sum_{j=1}^{d} (p_{ij}^2 - 2p_{ij}q_j) + \beta \sum_{j=1}^{d} (r_j + s_jt_j) + r_{d+1}\beta + s_{d+1}u. \end{aligned}$$

For all $i, h \in [m]$, we have

$$\begin{aligned}\boldsymbol{p}_i'(\boldsymbol{q}')^T - \boldsymbol{p}_h'(\boldsymbol{q}')^T &= \alpha\beta\left(\sum_{j=1}^{d}\left(p_{ij}^2 - 2p_{ij}q_j\right) - \sum_{j=1}^{d}\left(p_{hj}^2 - 2p_{hj}q_j\right)\right)\\ &= \alpha\beta\left(\sum_{j=1}^{d}\left(p_{ij} - q_j\right)^2 - \sum_{j=1}^{d}\left(p_{hj} - q_j\right)^2\right)\\ &= \alpha\beta\left(D\left(\boldsymbol{p}_i, \boldsymbol{q}\right) - D\left(\boldsymbol{p}_h, \boldsymbol{q}\right)\right).\end{aligned}$$

As α and β are positive, then, the equation (1) holds. □

Theorem 2: In the Outsourcing*k*-NNComputation of our scheme, the cloud server can correctly find out the *k*-NN of the query point $\boldsymbol{q}$ according to the distance $\boldsymbol{p}_i'(\boldsymbol{q}')^T$.

Proof: According to Theorem 1, the cloud server can exactly compare the distance between $\boldsymbol{q}$ and different tuples in $\mathbf{D}$ through using the value $\boldsymbol{p}_i'(\boldsymbol{q}')^T$.

Consequently, cloud server can correctly compute *k*-NN of $\boldsymbol{q}$ by linear scanning $\mathbf{D}'$ in our scheme, which completes the proof of Theorem 2. □

Table 1 Computation complexity

Scheme	*Stage*	*Data owner*	*Query user*	*Cloud server*	*Total*
ASPE (Wong et al., 2009)	KeyGen	$O(d^2)$	-	-	$O(d^2)$
	TupleEnc	$O(md^2)$	-	-	$O(md^2)$
	QueryEnc	-	$O(d^2)$	-	$O(d^2)$
	OutKNN	-	-	$O(md)$	$O(md)$
	Total	$O(md^2)$	$O(d^2)$	$O(md)$	$O(md^2)$
Our scheme	KeyGen	$O(d^2)$	-	-	$O(d^2)$
	TupleEnc	$O(md^2)$	-	-	$O(md)$
	QueryEnc	$O(d^2)$	$O(d^2)$	-	$O(d^2)$
	OutKNN	-	-	$O(md)$	$O(md)$
	Total	$O(md^2)$	$O(d^2)$	$O(md)$	$O(md^2)$

5.2 Cost analysis and comparison

Here, we analyse our scheme's computation complexity and communication overheads, and compare it with the existing efficient scheme ASPE (Wong et al., 2009).

The computation complexity of each stage of ASPE and our scheme is illustrated as Table 1. The only difference between them is that during query encryption, data owner in ASPE does nothing, but our scheme introduces new computing load to the data owner. The reason is that to resist against level-3 adversary, the data owner in our scheme will not directly reveal the key to query user, and he also can effectively manage each query, thus, data owner will burden extra $O(d^2)$ computational task. Generally speaking, d is not big in real-world applications, therefore, the extra computation load of data owner is still feasible.

Table 2 Communication overheads

Scheme	*TupleEnc*	*QueryEn*	*OutKNN*
ASPE (Wong et al., 2009)	$m(d+1)b_0$	$(d^2+2d+1)b_0$	$(d+1+k)b_0$
Our scheme	$m(2d+2)b_0$	$(2d^2+5d+2)b_0$	$(2d+2+k)b_0$

If each dimension or the index is b_0 bit, then the communication cost of ASPE (Wong et al., 2009) and our scheme is as Table 2 where the communication overheads of TupleEnc are spent on uploading the encrypted database to cloud server. The comparison result shows that communication overheads of our scheme are about as twice as that of ASPE (Wong et al., 2009), since we extend the d-dimensional original tuples into $(2d+2)$-dimensional encrypted ones for higher security.

5.3 Security analysis

The data privacy and query privacy are discussed as follows:

- *Query privacy*: For the privacy of query point $\boldsymbol{q}$, we will consider the view of data owner and cloud server, respectively.

  Throughout the k-NN computation, data owner receives nothing but $\{\dot{q}_1, \dot{q}_2, \cdots, \dot{q}_d\}$ from query user in which $\dot{q}_x = q_x + t_x$ (for $\forall x \in [d]$), as t_x is random to data owner, thus he cannot find out q_x.

  Cloud server can learn $\boldsymbol{q}'$ about the query point $\boldsymbol{q}$. As $\boldsymbol{q}' = \beta \boldsymbol{M}\dot{\boldsymbol{q}}^T$ where β and $\boldsymbol{M}$ are privately kept to data owner, cloud server cannot figure out the query point $\boldsymbol{q}$, either.

  In general, the query point can be well preserved while cloud server does not collude with data owner. As being mentioned in our foregoing design goals (Section 3.2), we do not consider the security of query points against the collusion attack of data owner and cloud server, because data owner and cloud server, by collusion, can inherently infer the query point with high accuracy from its k-NN, which is the output of the query.

- *Data privacy*: While executing k-NN query, the query user receives $\boldsymbol{X}$ and $\boldsymbol{Y}$, from which query user can derive out $(M_{j,2l} - M_{j,2l-1}) / (M_{x,2l} - M_{x,2l-1}) = Y_{jl} / Y_{xl}$ (only for j; $x \in [2d+2]$ and $l \in [d]$) about the matrix $\boldsymbol{M}_{(2d+2)\times(2d+2)}$ in the key of data owner $Key = \{\alpha, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{M}, \boldsymbol{M}^{-1}\}$.

  In the scheme, the encrypted database $\mathbf{D}' = \{\boldsymbol{p}_1', \boldsymbol{p}_2', \cdots, \boldsymbol{p}_m'\}$ is stored in cloud server which can also legally obtain the encrypted query point $\boldsymbol{q}'$. While cloud server (adversary) does not obtain $\boldsymbol{X}$ and $\boldsymbol{Y}$ that data owner sends to query users, our scheme is almost the same as the secure scheme against level-2 attack in Wong et al. (2009), and it has been shown that the data can be secure to resist an adversary who knows the encrypted tuples and some plain database points. Thus, our approach can preserve the database privacy under level-2 adversary model.

If some untrustworthy query users leak their knowledge about the key of data owner to cloud server, the outsourced database in ASPE (Wong et al., 2009) will be entirely broken. However, in our novel scheme, query users only learn $\boldsymbol{X}$ and $\boldsymbol{Y}$ but not the full key of data owner. Even when $\boldsymbol{X}$ and $\boldsymbol{Y}$ are revealed to cloud server by some unreliable query users, the cloud server still cannot deduce more information about $\boldsymbol{M}$ in $Key = \{\alpha, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{M}, \boldsymbol{M}^{-1}\}$, therefore, it cannot recover the database **D**. That is, our scheme can resist level-3 adversary. Additionally, we use a similar manner to encrypt the query points and the tuples of data owner, thus, the query points are also secure against the leakage of $\boldsymbol{X}$ and $\boldsymbol{Y}$.

6 Performance evaluation

This section further evaluates our scheme by experiments under Windows XP with Intel(R) Core(TM) i5 2.4GHz CPU and 2GB memory.

6.1 Key generation

We present the average running time of key generation of ASPE (Wong et al., 2009) and our new scheme in Figure 2. The results are averaged over 10^5 runs. It shows that our scheme costs more time than ASPE, but our scheme only spends less than 1 millisecond on key generation even when d is as high as 100. Thus, our time cost is still practical.

Figure 2 Average time of key generation

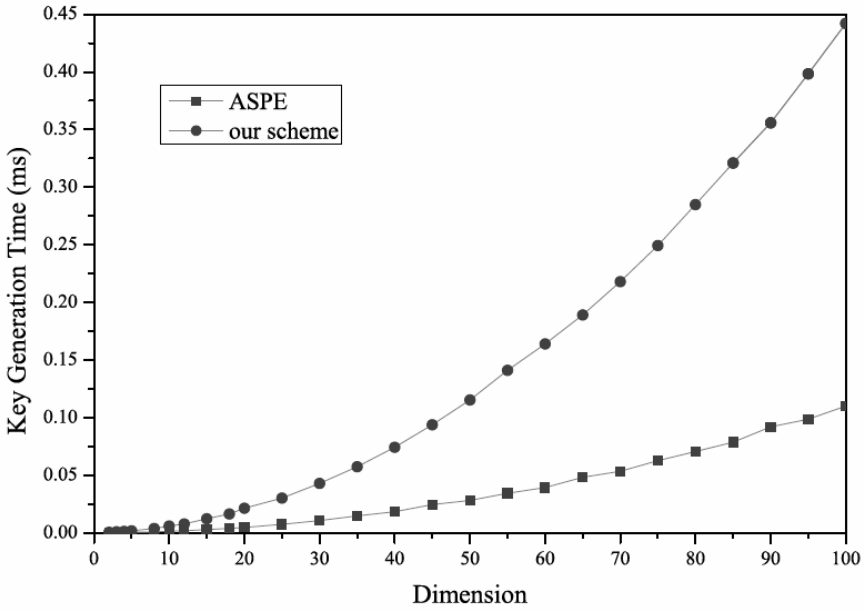
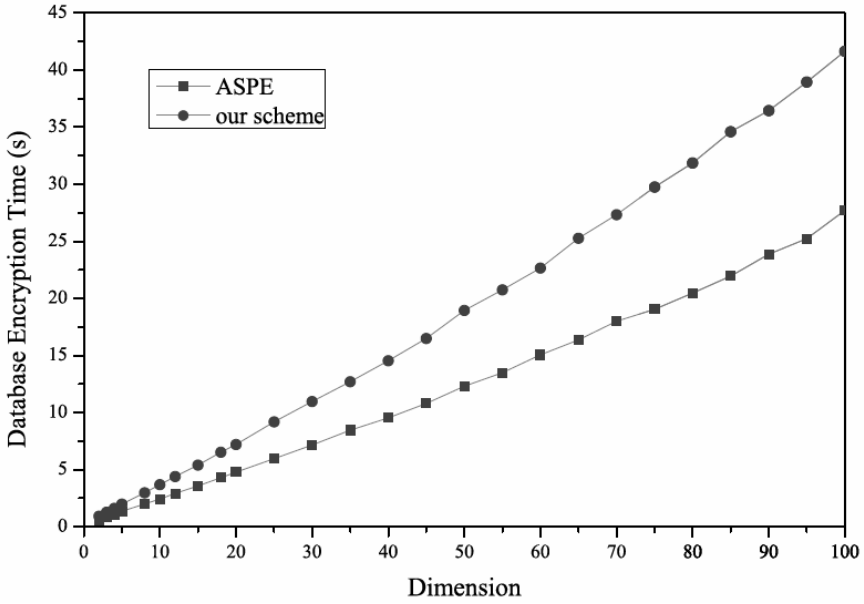
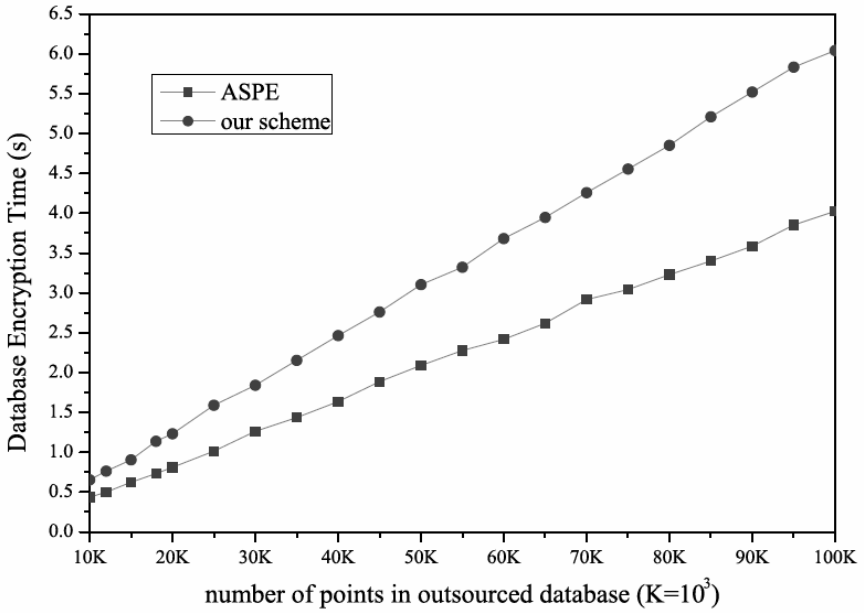


Figure 3 Running time of encrypting tuples, (a) m = 60 K (K= 10^3), d = 2 to 100 (b) d = 10, m = 10 K to 100 K



(a)



(b)

6.2 Database encryption

Figure 3(a) describes the results to encrypt the databases of 60 K points with dimension ranging from 2 to 100 through ASPE (Wong et al., 2009) and our scheme, and Figure 3(b) gives the running time of encrypting 10 K to 100 K ten-dimensional points. As can be seen, our scheme needs a little more time to complete tuples encryption, compared with the efficient previous scheme ASPE. However, the response time of tuples encryption is less stringent than that of query encryption and computation. While m = 100 K and d = 10, our scheme can finish database encryption within 6.5 seconds. For m = 60 K and d = 100, our database encryption time is less than 45 seconds. Therefore, the tuples encryption costs of our approach are relatively feasible in real-world applications.

6.3 Query encryption and outsourcing k-NN query

After query user has received the key $\boldsymbol{M}$ in ASPE (Wong et al., 2009), the query encryption time of ASPE is negligibly small. Compared with the negligible time cost in ASPE, our query encryption is time-consuming. Nevertheless, our scheme can complete encrypting one 100-dimensional query point within 1 second, which is acceptable for query encryption as well.

After receiving encrypted database and perturbed query point, cloud server, in ASPE and our scheme both, computes the k-NN tuples through linearly scanning the $\mathbf{D}'$. Their performances in this stage are similar to each other.

7 Conclusions

In this paper, we focused on the security and privacy of outsourcing database and k-NN query in cloud. Considering the problems that query users can access the full key to encrypt and decrypt the outsourced database, we proposed a new approach to protect the privacy of outsourced database and query points while supporting k-NN query on outsourced encrypted database. Compared with existing work, our scheme only releases partial information about the key to query users, and the data privacy can be preserved even some untrustworthy query users leak their knowledge about the key to adversary. Performance evaluations show our scheme is a little less efficient than the previous solution ASPE (Wong et al., 2009), but our new approach is still practical and feasible in real-world applications.

For the future work, we will devote to secure cloud storage and query schemes with better properties, such as completely preserving the key from anybody other than the data owner.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. The work is supported in part by Japan Society of the Promotion of Science (No. P12045) and the National Natural Science Foundation of China (No. 61272404).

References

Aggarwal, C.C. and Philip, S.Y. (2008) *Privacy-Preserving Data Mining: Models and Algorithms*, Vol. 34, Springer US, New York, NY, USA.

Amirbekyan, A. and Estivill-Castro, V. (2007) 'Privacy-preserving k-NN for small and large data sets', in *IEEE ICDM Workshops*, pp.699–704.

Cao, N., Wang, C., Li, M., Ren, K. and Lou, W. (2011a) 'Privacy-preserving multi-keyword ranked search over encrypted cloud data', in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pp.829–837.

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

Chen, K. and Liu, L. (2005) 'Privacy preserving data classification with rotation perturbation', in *5th IEEE International Conference on Data Mining (ICDM)*.

Chen, K., Sun, G. and Liu, L. (2007) 'Towards attack-resilient geometric data perturbation', in *SIAM Data Mining Conference*.

Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J. and Toft, T. (2006) 'Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation', in *3rd Theory of Cryptography Conference (TCC)*, LNCS, Vol. 3876, pp.285–304.

Ghinita, G., Kalnis, P., Kantarcioglu, M. and Bertino, E. (2011) 'Approximate and exact hybrid algorithms for private nearest-neighbor queries with database protection', *Geoinformatica*, Vol. 15, No. 4, pp.699–726.

Goethals, B., Laur, S., Lipmaa, H. and Mielikainen, T. (2004) 'On private scalar product computation for privacy-preserving data mining', in *7th International Conference on Information Security and Cryptology, LNCS*, Vol. 3506, pp.104–120.

Goldreich, O. (2004) *Foundations of Cryptography: Volume II, Basic Applications*, Cambridge University Press, Cambridge.

Goldreich, O., Micali, S. and Wigderson, A. (1987) 'How to play any mental game, or a completeness theorem for protocols with an honest majority', in *Proc. of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, ACM Press, pp.218–229.

Hashem, T., Kulik, L. and Zhang, R. (2010) 'Privacy preserving group nearest neighbor queries', in *13th International Conference on Extending Database Technology*, ACM, pp.489–500.

Hore, B., Mehrotra, S., Canim, M. and Kantarcioglu, M. (2012) 'Secure multidimensional range queries over outsourced data', *The VLDB Journal*, Vol. 21, No. 3, pp.333–358.

Hu, H., Xu, J., Ren, C. and Choi, B. (2011) 'Processing private queries over untrusted data cloud through privacy homomorphism', in *IEEE 27th International Conference on Data Engineering (ICDE)*, pp.601–612.

Kamara, S. and Lauter, K. (2010) 'Cryptographic cloud storage', in *Financial Cryptography: Workshop on Real-Life Cryptographic Protocols and Standardization, LNCS*, Vol. 6054, pp.136–149.

Kantarcioglu, M. and Clifton, C. (2004) 'Privately computing a distributed k-NN classifier', *8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pp.279–290.

Li, M., Yu, S., Lou, W. and Hou, Y.T. (2012) 'Toward privacy-assured cloud data services with flexible search functionalities', in *IEEE ICDCS Workshops*, pp.466–470.

Liu, K., Giannella, C. and Kargupta, H. (2006) 'An attacker's view of distance preserving maps for privacy preserving data mining', in *10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp.297–308.

Liu, K., Giannella, C. and Kargupta, H. (2008) 'A survey of attack techniques on privacy-preserving data perturbation methods', in *Privacy-Preserving Data Mining*, pp.359–381, Springer.

Oliveira, S.R.M. and Zaiane, O.R. (2003) 'Privacy preserving clustering by data transformation', in *Proc. of the 18th Brazilian Symposium on Databases*, pp.304–318.

Paillier, P. (1999) 'Public-key cryptosystems based on composite degree residuosity classes', in *EUROCRYPT*, *LNCS*, Vol. 1592, pp.223–238, Springer.

Papadopoulos, S., Bakiras, S. and Papadias, D. (2010) 'Nearest neighbor search with strong location privacy', *Proceedings of the VLDB Endowment*, Vol. 3, Nos. 1–2, pp.619–629.

Pedersen, T. (1991) 'Non-interactive and information-theoretic secure verifiable secret sharing', in *CRYPTO*, pp.129–140, Springer.

Qi, Y. and Atallah, M.J. (2008) 'Efficient privacy-preserving k-nearest neighbor search', in the *28th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp.311–319.

Shaneck, M., Kim, Y. and Kumar, V. (2006) 'Privacy preserving nearest neighbor search', in *IEEE ICDM Workshops*, pp.541–545.

Singh, M.D., Krishna, P.R. and Saxena, A. (2010) 'A cryptography based privacy preserving solution to mine cloud data', in the *3rd Annual ACM Bangalore Conference*.

Wang, C., Cao, N., Li, J., Ren, K. and Lou, W. (2010) 'Secure ranked keyword search over encrypted cloud data', in *30th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp.253–262.

Wang, C., Cao, N., Ren, K. and Lou, W. (2012a) 'Enabling secure and efficient ranked keyword search over outsourced cloud data', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, No. 8, pp.1467–1479.

Wang, C., Ren, K., Yu, S. and Urs, K.M.R. (2012b) 'Achieving usable and privacy-assured similarity search over outsourced cloud data', in *Proceedings of IEEE INFOCOM*, pp.451–459.

Wong, W.K., Cheung, D.W., Kao, B. and Mamoulis, N. (2009) 'Secure k-NN computation on encrypted databases', in *Proceedings of the 35th SIGMOD*, pp.139–152.

Xiong, L., Chitti, S. and Liu, L. (2007) 'Mining multiple private databases using a k-NN classifier', in *ACM Symposium on Applied Computing*, ACM, pp.435–440.

Xu, H., Guo, S. and Chen, K. (2012) 'Building confidential and efficient query services in the cloud with rasp data perturbation', *IEEE Transactions on Knowledge and Data Engineering*, December.

Yao, A.C. (1982) 'Protocols for secure computations', in the *23rd Annual IEEE Symposium on Foundations of Computer Science*, pp.160–164.

Yao, A.C. (1986) 'How to generate and exchange secrets', in *27th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp.162–167.

Yao, B., Li, F. and Xiao, X. (2013) 'Secure nearest neighbor revisited', in *29th IEEE International Conference on Data Engineering (ICDE)*.