sensors

MDPI

Article

Secure Nearest Neighbor Query on Crowd-Sensing Data

Ke Cheng [1], Liangmin Wang [2] and Hong Zhong [1,*]

[1] School of Computer Science and Technology, Anhui University, Hefei 230601, China; chengke@ahu.edu.cn
[2] School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China; wanglm@ujs.edu.cn
* Correspondence: zhongh@ahu.edu.cn; Tel.: +86-138-5515-1002

Academic Editor: Yike Guo
Received: 19 July 2016; Accepted: 14 September 2016; Published: 22 September 2016

Abstract: Nearest neighbor queries are fundamental in location-based services, and secure nearest neighbor queries mainly focus on how to securely and quickly retrieve the nearest neighbor in the outsourced cloud server. However, the previous big data system structure has changed because of the crowd-sensing data. On the one hand, sensing data terminals as the data owner are numerous and mistrustful, while, on the other hand, in most cases, the terminals find it difficult to finish many safety operation due to computation and storage capability constraints. In light of they Multi Owners and Multi Users (MOMU) situation in the crowd-sensing data cloud environment, this paper presents a secure nearest neighbor query scheme based on the proxy server architecture, which is constructed by protocols of secure two-party computation and secure Voronoi diagram algorithm. It not only preserves the data confidentiality and query privacy but also effectively resists the collusion between the cloud server and the data owners or users. Finally, extensive theoretical and experimental evaluations are presented to show that our proposed scheme achieves a superior balance between the security and query performance compared to other schemes.

Keywords: secure nearest neighbor; crowd-sensing; privacy-preservation; secure two-party computation; collusion attack

1. Introduction

Along with the popularization of mobile Internet and Internet of Things, a large quantity of ordinary users and sensor nodes have become involved in the perception and collection activities around the state of the environment. Hence, brand-new crowd sensing data emerge as the times require, and researchers are beginning to be concerned about the influence of such data on human life [1–5], including medical treatment, social networks, environmental monitoring, transportation, etc. The sensor data may contain private user details, especially for sensors that can collect location coordinates for Location Based Service (LBS). The cloud party has brought vast amounts of sensitive data together after data owners outsource their databases to the cloud server provider. Therefore, the inappropriate use of crowd sensing data, which not only contain user locations but also personality habits, health condition, social status and other sensitive information, brings great challenges to data confidentiality and user privacy [6–8].

To protect the confidentiality of the location data in the cloud, one straightforward way is to encrypt data by the data owner (Owner) before outsourcing. In addition, to preserve user privacy, authorized users (Users) need to perform a complex series of encryption and decryption operations during query execution. However, this approach cannot be directly applicable to crowd sensing data because the mobile terminals in crowd sensing networks fail to perform the current big computation limited to compute and storage capability. More importantly, mobile terminals, which are the source

of crowd sensing data, are mutually-distrusting as data owners. This situation makes up a totally different service structure, as depicted in Figure 1. We call it Multi Owners and Multi Users (MOMU) cloud services structure based upon crowd sensing data, referred to as MOMU structure. It is different from the traditional Single Owner and Multi Users (SOMU) structure portrayed in Figure 2, in which only one data owner has a large number of data and outsources them to the cloud, then authorized users access those data for issuing queries.

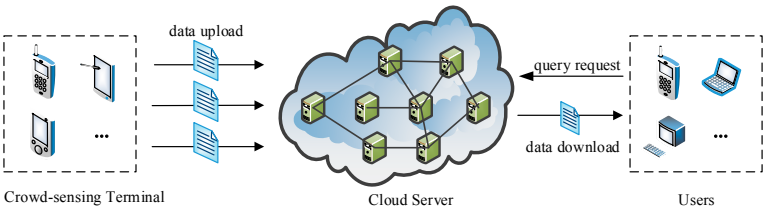

Figure 1. Multi Owners and Multi Users (MOMU) structure based upon crowd sensing data.

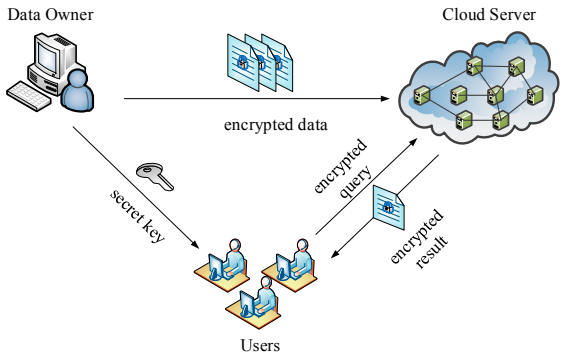

Figure 2. Traditional (Single Owner and Multi Users) SOMU model.

In this paper, we focus on the secure nearest neighbor (SNN) problem on crowd-sensing location data (MOMU structure is a typical structure in the applications of crowd sensing [9,10]), since LBS is the current hot topic in the study of big data [11–13], furthermore, nearest neighbor (NN) queries are fundamental in LBS [14,15]. In the past few years, researchers have proposed various methods [15–19] to address the SNN problem in SOMU model. The work in [16] uses a new encryption scheme (ASPE) to preserve scalar product between the query vector and any vector for distance comparison, which is sufficient to find NN. Hu et al. [17] propose a solution based on privacy homomorphism encryption scheme (ASM-PH). Instead of finding exact NN, [15] allows a cloud party to approximate it based on secure Voronoi diagram (SVD). Similar to [15], the work in [18] also uses Voronoi to raise efficiency. Elmehdwi et al. [19] propose a novel protocol over encrypted data based on Paillier cryptosystem [20], which can calculate encrypted distance between data record and query record in a secure way.

One important observation about these prior works is that the data owners are all assumed to be a single trusted party. Hence, in the MOMU structure, it is impractical to share the secret key between all the data owners and users just like existing solutions [15–19] because the compromise of any data owner would be a threat to data security of other owners. For instance, in a cloud system based on key-sharing, if an owner colludes with the cloud, the other owners' data stored in cloud will be leaked because they could be decrypted with a sharing key. A natural idea is that multiple data owners could use their own unique keys. However, the SNN query across the data encrypted by different keys is another challenge (e.g., data availability, key management, etc.). In addition, the mobile terminals in crowd sensing networks cannot fulfill the requirements for computation and storage capability of the end-user in traditional methods. Therefore, the methods based on SOMU structure cannot be applied to crowd-sensing cloud server directly.

To address those challenges, our insight is that there is generally a proxy server of service providers in a cloud environment. Thus, we can use the proxy server to share the hard work for the end-user. In order to ensure availability of encrypted data by different keys, we also provide a series of protocols of secure two-party computation coordinating to the proxy architecture, which not only protects the confidentiality of the location data from various data owners but also allows the specified user to perform the SNN query efficiently. In summary, our paper makes the following contributions:

- We propose a Security Architecture over MOMU Cloud Service System (SAMOMU) model based on partition of public cloud and proxy server to meet the security and performance requirements of MOMU structure.
- In the SAMOMU model, a method to solve the SNN problem is presented by combining SVD method and a series of secure two-party computation protocols.
- We present an extensive experimental evaluation of the proposed scheme, which shows that the proposed method has good performance for crowd-sensing data.

The remainder of this paper proceeds as follows. Related works are surveyed in Section 2. We define our system model and design goals in Section 3. A set of basic security protocols which are utilized in our scheme are provided in Section 4. Section 5 presents the details of our scheme. The security and performance analysis are carried out in Section 6. Finally, Section 7 concludes the paper and discusses potential future directions.

2. Related Works

In this section, we first review several nearest neighbor query methods for location privacy in LBS, and then we present an overview of the existing SNN techniques.

2.1. Query Location Privacy in LBS

In traditional LBS model, the methods should ensure location privacy in the sense that the user does not reveal any information about his location to the LBS provider. In this case, LBS server acts as the role of data owner. As a consequence, there is a simpler security requirement compared with the SNN query in the cloud, which focuses mainly on privacy preserving for the users.

In general, several main techniques for location privacy have been investigated in current studies. The first is the cloaking regions method [21,22], which assumes a trust anonymous party between the user and the server for transforming actual locations into vague locations. Obviously, the anonymizer becomes a communication bottleneck and a vulnerable point of attack. To count this privacy attack, Gao et al. [23] propose a distributed structure for location privacy protection without a centralized anonymous server. Another category of work relies on Private Information Retrieval (PIR) [24] to provide strong location privacy. This technique allows users to retrieve an object stored by a server without revealing which record he is retrieving. However, these PIR-based solutions [25,26] are still not efficient enough to be implemented on a real system.

2.2. Existing SNN Techniques

Existing SNN techniques generally rely on SOMU model, which only contains a single trusted data owner, as depicted in Figure 2. Compared to the MOMU model, the significant difference is: the MOMU model involves multiple mutually-distrusting data owners.

In the methods [15–19] based on SOMU model, the data owner outsources his database and DBMS functionalities (e.g., NN query) to the cloud server providers where only trusted users are allowed to query the host data. Wong et al. [16] proposed a new encryption scheme (ASPE) that preserves the relative distances of all the database point to any query point that is sufficient to find NN. ASPE transforms data points and queries with secret matrices, which are symmetric keys for the encryption scheme. Thus, it must be shared with both the data owner and query users. As an alternate, Hu et al. [17] proposed a method based on Privacy Homomorphism (ASM-PH) encryption

scheme. During query processing, data owner sends the encrypted shadow index to user, and user needs to traverse the index locally to compute the distance between query point and an indexed point with the help of server. However, the methods in [16,17] are not secure because they are prone to chosen-plaintext attacks [15].

To further improve the query performance, Yao et al. [15] designed a novel SNN method based on secure Voronoi diagram (SVD). Instead of return exact NN, they allow a cloud server to return a relevant data partition. What is more notable, is that the work in [18] also used Voronoi and order-preserving encryption (OPE) to solve the SNN problem accurately. Although it can provide exact result, the solution incurs expensive overhead of computation and communication on the end-user. More importantly, the encryption schemes used in [15,18] are symmetric, and both the data owners and users have to share the secret key, which make it impractical in MOMU structure where there are multiple mutually-distrusting data owners.

Recently, Elmehdwi et al. [19] proposed a number of novel protocols over encrypted data based on Paillier cryptosystem [20], which can further increase security during query execution. They assume the existence of two semi-honest cloud servers P_1 and P_2 such that the encrypted data is known only to P_1, whereas the secret key is just revealed to P_2. Using the secure protocols, P_1 collaborate with P_2 for the final result after receiving an encrypted query from the user. However these protocols cannot be put into use for inefficiency.

Crowd-sensing cloud server is based on the MOMU structure, in which the number of data owners increases and computing power of end-users decreases compared with SOMU structure. These changes about objective conditions cause the changes of the security and performance requirements. Hence, the methods above do not apply to MOMU structure in which there are multiple mutually-distrusting and the end-user cannot afford huge costs for compute or storage.

3. System Model and Design Goals

In this section, we formalize the system model, security and privacy requirements, and describe our design goals.

3.1. System Model

The cloud service system based upon crowd-sensing data is actually aggregations of the crowd-sensing system and the cloud system. The terminals in this system are divided into two kinds of entities in the function: the data owner (Owner) and the data user (User). As a data owner, the terminal will outsource his/her data to the cloud for efficient storage and management. In fact, there is generally a proxy server of service providers in a cloud service based on crowd-sensing data. With the crowd-sensing data in VANET, for example, the data collected through VANET are uploaded to the cloud and governed by the traffic administrative department while the users such as automobile manufacturers, garages and insurance companies need to access the relevant data. Nowadays, large companies usually set up their own proxy server for different types of server. In this scenario, the traffic administrative department can be viewed as a trusted authority (TA). When a user wants to check the information about insurance and vehicle maintenance, he has to access them using the proxy of the insurance company and the garage, respectively. This is similar to the social network, which may contain a variety of services in regard to foods, sports, garments and so on, a user acquires different kinds of data through the corresponding proxy servers of service providers. To this end, we propose a Security Architecture over MOMU Cloud Service System (SAMOMU), as depicted in Figure 3.

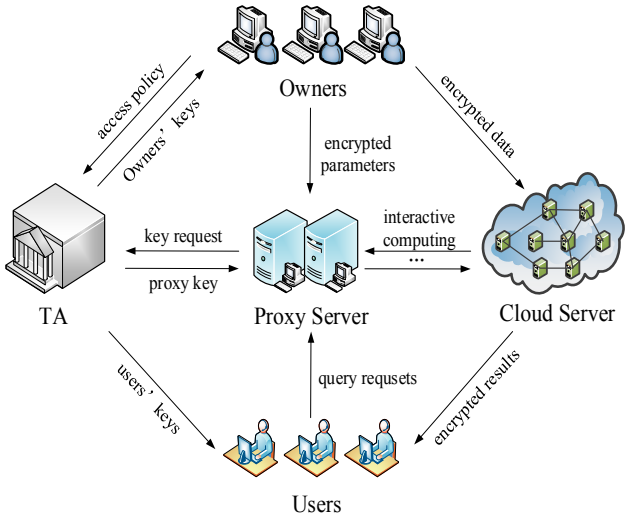

Figure 3. Security Architecture over MOMU Cloud Service System (SAMOMU).

Our system consists of five types of generic entities: Data Owners (Owners), Data Users (Users), Cloud Server (CS), Proxy Server (PS) and Trusted Authority (TA).

(1) Data Owners (Owners): We assume that our system requires m Owners, which are generally played by the sensor nodes, mobile phones and vehicle terminals, denoted as Own = $\{O_1, O_2, \ldots, O_m\}$. Data are generated by Owners, encrypted using their secret key and then outsourced to CS for storage.

(2) Data Users (Users): The system assumes there are n Users with limited computation and communication resources, denoted as Usr = $\{U_1, U_2, \ldots, U_n\}$. Note that the terminals that act as Owners could also take up positions as Users. They forward queries to the CS via the PS for the nearest neighbor.

(3) Cloud Server (CS): The CS stores all the encrypted data outsourced from Owners. After received NN requests from the PS, the CS interacts with the PS to process the data and returns the results to the Users.

(4) Proxy Server (PS): The PS takes on the task of providing those users with proxy services. In reality, much useful information is distributed among the crowd sensing networks, hence the PS normally caches the parsing results or extracts the metrics of interest. In SAMOMU model, the PS will host part of the computing task for Users.

(5) Trusted Authority (TA): TA is assumed to be trusted by all the other entities in the system to distribute and manage all the private keys, and to generate some parameters involved in the system.

Note that our systems are scalable and efficient for users. Specifically, users do not need to know the identities of other users or the total number of users involved in computation. Most importantly, because of the PS, the computation is non-interactive to users—users only need to outsource encrypted data initially and remain offline until retrieving encrypted outputs. It has been proven that the traditional single server model for secure outsourced computation cannot completely eliminate interactions between the user side and the server side (due to the impossibility of program obfuscation). The defect of this architecture is that the PS is likely to become a Single Point of Failure (SOF). However, in the real world, all service providers have the separated proxy server, which is totally independent of each other. Furthermore, service providers can adopt the hot-standby technique for solving the SOF from a view of engineering. Although the providers need to increase investment in infrastructure, it would make for a pleasant user experience in return. This is also the basic motivation of the paper.

3.2. Security and Privacy Requirements

In our security and privacy model, we assume PS and CS are both semi-honest (i.e., honest-but-curious). Meanwhile, we also assume these two servers are non-colluding. It means that neither of these two servers intends to corrupt users' data or computation process to prevent users from utilizing data correctly, but each server will try to learn the content of users' data (i.e., inputs), intermediate or final results of the computation without colluding with another server.

We remark that those assumptions are not initiated by our work, but rather derive from the related research [19,27–29]. According to the requirements of crowd-sensing scenario, the SAMOMU partitions server functions under the management of the TA. Actually, the security of our system is stronger than the Two-Clouds architecture [19], because the TA would be charge of the key management, the collusion between the PS and the CS cannot breakdown the full security of our system. To provide a flexible tradeoff between security and performance, we define the concrete data confidentiality and query privacy to against adversary *Adv* as follows.

Definition 1 (Data Confidentiality Definition). *Upon completion of the SAMOMU model, Adv cannot learn any plain data stored in the CS when Adv did not collude with any Owners. If an Owner was captured by Adv, the adversary would not get any assistance to obtain sensor data generated by other Owners.*

Definition 2 (Query Privacy Definition). *Neither the query point nor the result for users should be reveal to the Adv.*

To satisfy these privacy requirements, the active adversary *Adv* in our model has the following attacking abilities: *Adv* may eavesdrop all the communication links to get the encrypted data. In addition, *Adv* may compromise CS, some Users and Owners simultaneously, but subjects to the following restrictions: (1) *Adv* cannot compromise the CS and the PS at the same time; and (2) in a process of query, *Adv* cannot compromise the User who launched this query. Moreover, we do not aim to protect access pattern in this paper due to the extremely high complexity, i.e., to protect it, the algorithm has to "touch" the whole dataset [24].

3.3. Design Goals

In order to achieve the SNN query under SAMOMU model, our method will fulfill privacy and performance guarantees as follows:

- Data confidentiality and query privacy: The data confidentiality and query privacy as described in the Definitions 1 and 2 should be guaranteed.
- Reduce the end-users' cost: The end-users in SAMOMU model generally have limited computation and communication resources, thus our method should be designed for reducing the end-users' cost by using the PS efficiently.
- Access Control: A large number of parties are involved in the system, therefore control of the user's access request by attribute-based encryption (ABE) [30] is necessary.

We list the main technologies used in our method in Table 1; these cannot apply to our method directly, and the improvements and combinations of them are technical contribution of our work.

Table 1. Requirements and Key Techniques under SAMOMU model.

Requirements	Key Techniques
Data confidentiality and query privacy	SVD method and the encryption based on secret-sharing
Reduce the end-users' cost	secure two-party computation protocols
Access Control	attribute-based encryption

4. Basic Security Protocols

In this section, we present a set of secure two-party computation protocols that will be used as sub-routines while constructing our proposed scheme in Section 5. We firstly introduce an encryption scheme using secret-sharing [31], based here to build our protocols.

4.1. The Encryption Scheme Based on Secret-Sharing

Under secret sharing, the encryption scheme used in [31] aims to split a plaintext into a secret key and a ciphertext for data confidentiality. The concrete algorithm is showed in Definition 3.

Definition 3. *The secret sharing encryption process consists of two steps:*

Step 1 (Key Generation). *Generate a public parameter PP = <g, n> in the follow way: choose randomly two prime numbers p and q, then compute* $n = p \times q$, $\varphi(n) = (p-1)\cdot(q-1)$. *Choose randomly a positive number g that is co-prime with n. Generate randomly a secret key sk = {m, a} (0 < m, a < n).*

Step 2 (Share Computation). *Given a sensitive value x, choose randomly a number r, the encrypted value* $E_{sk,r}(x)$ *is given by* $E_{sk,r}(x) = x\cdot(mg^{ra} \bmod n)^{-1} \bmod n$, *where* $(\,)^{-1}$ *denotes the modular inversion. To recover x, one needs all shares sk, r and* $E_{sk,r}(x)$ *and compute* $D_{sk,r}(E_{sk,r}(x)) = E_{sk,r}(x)\cdot(mg^{ra} \bmod n) \bmod n$. *We refer the reader to [31] for correctness and security proof of this scheme.*

4.2. Secure Two-Party Computation Protocols

We present a set of protocols based on the encryption scheme above. All of the below protocols are considered under two-party semi-honest setting: Data Normalization (DataNorm) protocol, Secure Distance (SecDist) protocol, Secure Compare (SecComp) protocol, Secure Minimum of k Numbers (SecMin_k) protocol.

Data Normalization (DataNorm). We assume that a party P_1 holds a secret key $sk_1 = \{m_1, a_1\}$, a random number r_1, a target key $sk_2 = \{m_2, a_2\}$ and a target number r_2 while a party P_2 has encrypted value $E_{sk_1,r_1}(x)$. The goal of the DataNorm protocol is to compute the encryption of x, which is encrypted by sk_2 and r_2. At the end, the output is known only to P_2. In our query scheme described in Section 5, we will use the DataNorm protocol to make a data normalization over the encrypted data, although those data were encrypted using different keys of multiple data owners. Thanks to this, we can ensure availability of encrypted data. The protocol is shown in Algorithm 1.

Algorithm 1. DataNorm $(E_{sk_1,r_1}(x), \{sk_1,r_1\}, \{sk_2,r_2\}) - E_{sk_2,r_2}(x)$

Require: P_1 has $sk_1 = \{m_1, a_1\}$, $sk_2 = \{m_2, a_2\}$,r_1, r_2; P_2 has $E_{sk_1,r_1}(x)$

(1) P_1:
- (a) Pick two random numbers m_3, a_3, r_3
- (b) $p \leftarrow {a_3}^{-1}(r_2a_2 - r_1a_1) \bmod \varphi(n)$
- (c) $q \leftarrow m_1{m_3}^p{m_2}^{-1} \bmod n$
- (d) $s \leftarrow (m_3g^{a_3} \bmod n)^{-1}$
- (e) Send p, q, s to P_2

(2) P_2:
- (a) $E_{sk_2,r_2}(x) \leftarrow E_{sk_1,r_1}(x)\cdot q\cdot s^p$

Definition 4 (Correctness). *If DataNorm protocol presented in Algorithm 1 is correct, a party* P_2 *can get the encryption of x, which is encrypted by* sk_2 *and* r_2.

Proofs of Correctness. We can use sk_2 and r_2 to decrypt ciphertext of P_2, converting from $E_{sk_2,r_2}(x)$ back to plain text x. The process is as follows.

$$\begin{aligned} D_{sk_2,r_2}(E_{sk_2,r_2}(x)) &= E_{sk_1,r_1}(x) \cdot q \cdot s^p \cdot m_2 \cdot g^{r_2 a_2} \\ &= E_{sk_1,r_1}(x) \cdot m_1 \cdot {m_3}^p \cdot {m_2}^{-1} \cdot s^p \cdot m_2 \cdot g^{r_2 a_2} \\ &= E_{sk_1,r_1}(x) \cdot m_1 \cdot {m_3}^p \cdot ((m_3 \cdot g^{a_3})^{-1})^p \cdot g^{r_2 a_2} \\ &= E_{sk_1,r_1}(x) \cdot m_1 \cdot (g^{a_3 \cdot ({a_3}^{-1}(r_2 a_2 - r_1 a_1))})^{-1} \cdot g^{r_2 a_2} \\ &= E_{sk_1,r_1}(x) \cdot m_1 \cdot g^{r_1 a_1} = x \end{aligned}$$

□

Secure Distance (SecDist). Consider a party P_1 with secret key sk, a secret share r and a party P_2 with private input $E_{sk,r}(\boldsymbol{X})$, $E_{sk,r}(\boldsymbol{Y})$. Here, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are two-dimensional vectors where $E_{sk,r}(\boldsymbol{X}) \leq E_{sk,r}(x_1)$, $E_{sk,r}(x_2)>$, and $E_{sk,r}(\boldsymbol{Y}) = <E_{sk,r}(y_1)$, $E_{sk,r}(y_2)>$. The goal of the SecDist protocol is to compute $E_{sk,r}(|X-Y|^2)$, where $|X-Y|^2$ denotes the Euclidean distance between X and Y. During this protocol, no information regarding X and Y is revealed to P_1 and P_2. The SecDist protocol described in Algorithm 2 will be used as a sub-routine to construct our SNN method in Section 5.

Algorithm 2. SecDist($E_{sk,r}(X)$,$E_{sk,r}(Y)$) $-$ $E_{sk,r}(|X-Y|^2)$

Require: P_1 has $sk = \{m, a\}$, r; P_2 has $E_{sk,r}(|X-Y|^2)$

(1) P_2:

(a) $E_{sk',r}(|X-Y|^2) \leftarrow (E_{sk,r}(x_1) - E_{sk,r}(y_1))^2 + (E_{sk,r}(x_2) - E_{sk,r}(y_2))^2$

(2) P_1:

(a) $m' \leftarrow m^2$, $a' \leftarrow 2a$, $sk' \leftarrow \{m', a'\}$

(3) P_1 and P_2:

(a) $E_{sk,r}(|X-Y|^2) \leftarrow$ DataNorm($E_{sk',r}(|X-Y|^2)$,$\{sk', r\}$,$\{sk, r\}$)
(b) P_2 get $E_{sk,r}(|X-Y|^2)$

Definition 5 (Correctness). *If SecDist protocol presented in Algorithm 2 is correct, a party P_2 can get the value $E_{sk,r}(|X-Y|^2)$, which can be decrypted by sk and r.*

Proofs of Correctness. We can use sk and r to decrypt ciphertext of P_2, converting from $E_{sk,r}(|X-Y|^2)$ back to plain text $|X-Y|^2$. The process is as follows.

$$\begin{aligned} D_{sk,r}(E_{sk,r}(\left|X-Y\right|^2)) &= D_{sk',r}(E_{sk',r}(\left|X-Y\right|^2)) \\ &= D_{sk',r}((x_1 \cdot (m \cdot g^{ra})^{-1} - y_1 \cdot (m \cdot g^{ra})^{-1})^2 \\ &\quad + (x_2 \cdot (m \cdot g^{ra})^{-1} - y_2 \cdot (m \cdot g^{ra})^{-1})^2) \\ &= D_{sk',r}(((x_1-y_1)^2 + (x_2-y_2)^2) \cdot (m^2 \cdot g^{2ra})^{-1}) \\ &= ((x_1-y_1)^2 + (x_2-y_2)^2) \cdot (m^2 \cdot g^{2ra})^{-1} \cdot m^2 \cdot g^{2ra} \\ &= (x_1-y_1)^2 + (x_2-y_2)^2 \end{aligned}$$

□

Secure Compare (SecComp). In this protocol, P_1 holds $sk = \{m, a\}$, r and P_2 holds $E_{sk,r}(x)$, $E_{sk,r}(y)$. The goal of the SecComp protocol is to compare x with y without revealing any information about x

and y to P_1 and P_2. This protocol returns true if $x > y$, otherwise it returns false. The protocol is shown in Algorithm 3.

Algorithm 3. SecComp($E_{sk,r}(x)$, $E_{sk,r}(y)$) $\rightarrow$ true/false.

Require: P_1 has $sk = \{m, a\}$, r;P_2 has $E_{sk,r}(x)$ and $E_{sk,r}(y)$

(1) P_1:

(a) Pick a random positive number z
(b) $E_{sk,r}(z) \leftarrow z \cdot m \cdot g^{ra}$
(c) Send $E_{sk,r}(z)$ to P_2

(2) P_2:

(a) $t \leftarrow (E_{sk,r}(x) - E_{sk,r}(y)) \cdot E_{sk,r}(z)$

(3) P_1:

(a) $sk' = \langle m^2, 2a \rangle$,$sk'' = \langle 1, 0 \rangle$

(4) P_1 and P_2:

(a) $f \leftarrow$ DataNorm(t,sk',sk'')

(5) P_2:

(a) **IF** $f > 0$ **THEN** **RETURN** true**ELSE** **RETURN** false **RETURN** false

Definition 6 (Correctness). *If SecComp protocol presented in Algorithm 3 is correct, a party P_2 will get a $f > 0$ iff $x > y$.*

Proofs of Correctness. $f = E_{sk'',r}((x - y) \cdot z) = (x - y) \cdot z \cdot 1 \cdot g^{r \cdot 0} = (x - y) \cdot z$ and $z > 0$, so iff $f > 0$ then $x > y$, else $x < y$. □

Secure Minimum of k Numbers (SecMin$_k$). We assume that P_1 has $sk = \{m, a\}$, r and P_2 has $E_{sk,r}(x_1)$, $E_{sk,r}(x_2)$, ... , $E_{sk,r}(x_k)$, the goal of the SecMin$_k$ protocol is to securely compute $Min = \min(x_1, x_2, \ldots, x_k)$. During this protocol, no information regarding x_i ($1 \leq i \leq k$) is revealed to P_1 and P_2. On the basis of the SecComp protocol, all the values in the SecMin$_k$ protocol are compared in pairs using the divide-and-conquer strategy. Note that the computation complexity of SecMin$_k$ is bounded by $O(\log_2 k)$. For instance, P_1 has $sk = \{m, a\}$, r and P_2 has $E_{sk,r}(x_1)$, $E_{sk,r}(x_2)$, ... , $E_{sk,r}(x_6)$, the minimum value solving process is present in Figure 4. The protocol is shown in Algorithm 4.

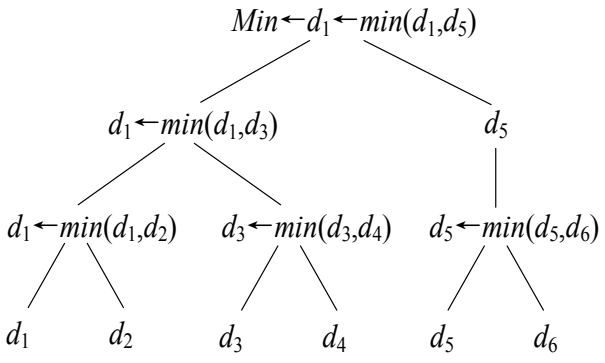

Figure 4. The minimum value solving process by SecMin$_k$ for $k = 6$.

Algorithm 4. $\text{SecMin}_k(E_{sk,r}(x_1), E_{sk,r}(x_2), \ldots, E_{sk,r}(x_k)) \rightarrow Min.$

Require: P_1 has $sk = \{m, a\}$, r; P_2 has $E_{sk,r}(x_1), E_{sk,r}(x_2), \ldots, E_{sk,r}(x_k)$

(1) P_2:

(a) $d_i \leftarrow E_{sk,r}(x_i)$, FOR $1 \leq i \leq k$

(b) $num \leftarrow k$

(2) P_1 and P_2, **FOR** $i = 1$ **TO** $\lceil \log_2 k \rceil$:

(a) **FOR** $1 \leq j \leq \lfloor num/2 \rfloor$:

IF $i = 1$ **THEN**
IF SecComp(d_{2j-1}, d_{2j}) **THEN** $d_{2j-1} \leftarrow d_{2j}$
ELSE
IF SecComp$(d_{2i(j-1)+1}, d_{2ij-1})$ **THEN** $d_{2i(j-1)+1} \leftarrow d_{2ij-1}$

(b) $num \leftarrow \lceil num/2 \rceil$

(3) P_1:

(a) $Min \leftarrow d_1$

5. The Proposed SNN-SAMOMU Query Scheme

Based on the secure two-party computation protocols presented in Section 4, we propose a SNN query scheme in SAMOMU model, which consists of the following phases: System Setup, Data Outsourcing, Access Control and Result Query. Figure 5 shows a SNN-SAMOMU query framework. Firstly, TA initializes the system, then data owners encrypt their data and outsource the corresponding encrypted data to CS while uploading random parameters to PS. To guarantee the access control, data owners use attribute-based encryption (ABE) to encrypt their own secret keys and send them to TA for management. Once the data user is authenticated by TA, PS will receive a proxy key from TA for computation. In the result query phase, PS will cooperate with CS to perform a query protocol for a result point as output to the user. Finally, we present two strategies to boost performance of our scheme.

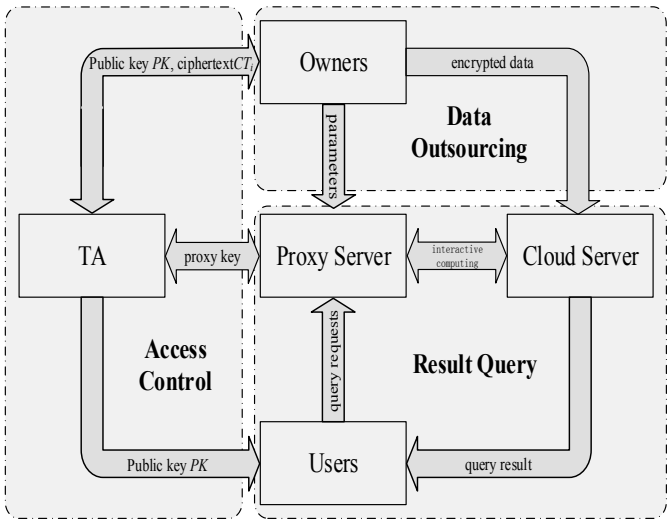

Figure 5. The query framework based on SNN-SAMOMU model.

5.1. System Setup

The TA calls the Key Generation algorithm to generate a public parameter *PP*, the users' keys for *m* Users and the owners' keys for *n* Owners. Let Key_O_i ($1 \leq i \leq m$) and Key_U_j ($1 \leq j \leq n$) denote users' keys and owners' keys, respectively. The TA publishes *PP* and sends the keys to the corresponding Owners and Users via secure channels.

5.2. Data Outsourcing

The Owners divide the data space into *K* disjoint intervals through SVD algorithm [15] locally, then use the *PP* and owners' keys to encrypt their own data and index by the encryption scheme described in Section 4.1. Finally, the encrypted data and index are outsourced to the CS. Our data outsourcing protocol runs in the following four steps.

(1) The data owner O_i receives a public parameter *PP* and his key Key_O_i.

(2) O_i divides the data space, which is corresponding to his two-dimensional point set D_i, into K_i disjoint intervals through SVD algorithm, then obtains K_i rectangular data partition $B_{i,k}$ presented in Figure 6, i.e., $D_i = <B_{i,1}, B_{i,2}, \ldots, B_{i,K_i}>$. Obviously, the rectangular partition can be uniquely identified by its lower-left (*LL*) and upper-right (*UR*) corners.

(3) O_i randomly select a number r_o_i and encrypt K_i data partition above through the using of Key_O_i and r_o_i, then obtains K_i data items in the format shown in Figure 7. The process of encryption is described in Algorithm 5.

(4) O_i uploads the data items generated in Step 3 to the CS and send r_o_i to the PS.

Algorithm 5. BlockEncryption.

Input:Key_O_i, r_o_i, K_i data partition.
Output:K_i data items in the format shown in Figure 7.

(a) **WHILE**($1 \leq k \leq K_i$)

(b) Encrypt the *LL* and *UR* of $B_{i,k}$ to get $E_{Key\_O_i,r\_o_i}(x_{LL})$, $E_{Key\_O_i,r\_o_i}(y_{LL})$, $E_{Key\_O_i,r\_o_i}(x_{UR})$, $E_{Key\_O_i,r\_o_i}(y_{UR})$

(c) Encrypt the points contained within the scope of $B_{i,j}$ to get $\{E_{Key\_O_i,r\_o_i}(t)\}_{t \in B_{i,k}}$, where $E_{Key\_O_i,r\_o_i}(t) = \langle E_{Key\_O_i,r\_o_i}(x_t), E_{Key\_O_i,r\_o_i}(y_t) \rangle$

(d) construct the data item in the format shown in Figure 7

(e) **END WHILE**

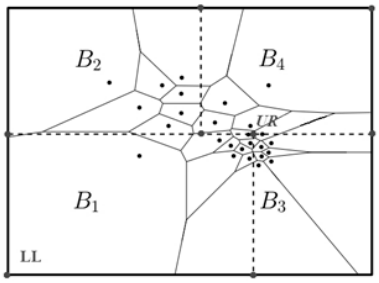

Figure 6. The secure Voronoi diagram (SVD) method for $K = 4$.

$ID_{i,k}$	$E_{Key\_O_i,\ r\_o_i}(x_{LL})$	$E_{Key\_O_i,r\_o_i}(y_{LL})$	$E_{Key\_O_i,r\_o_i}(x_{UR})$	$E_{Key\_O_i,r\_o_i}(y_{UR})$	$\{E_{Key\_O_i,r\_o_i}(t)\}_{t\in B_{i,k}}$

Figure 7. Format of outsourced data.

5.3. Access Control

In our method, the users have the capacity to access the encrypted data on the CS via the PS, which are uploaded by the Owners. In the real scenario, however, not all of the data can be visited by all users, only the user who was authenticated by the data owner can access the uploaded data. Hence the access policy in our system is necessary. In this paper, we use ABE [30] to achieve access control in which the data owner has the right to set access policy, so it is suitable for the data-sharing of crowd sensing networks.

For example, in VANET, a data collector as the data owner will outsource their data to the cloud, but these data are only expected to open to the owners of the A-region and the B-car. Naturally, he informs the management department as the TA of the access condition. Before owners visiting the data stored in cloud through a proxy server of the manufacturer, the proxy needs to send the owner's attributes (area, automaker, etc.) to management department for the permission to the specific dataset. Owners cannot visit the data in the cloud via the proxy until the condition is met.

The framework of access control is shown in Figure 8. All data owners upload their ciphers that contain access policy to TA. After receiving a query request from the User, PS sends the user's attributes to TA to be verified and obtains the proxy key. Once being verified by TA, PS can obtain the proxy key and perform the next phase of the query over the corresponding dataset in the cloud. The protocol sequence diagram in access control phase is shown in Figure 9. Specific processes are as follows:

(1) TA generates the public key PK and the master key MK used in ABE and publishes the PK.
(2) The data owner O_i with a access policy AP_i, PK and Key_O_i computes ciphertext CT_i and sends it to TA.
(3) The data user U_j sends his own attributes Ω_j to PS.
(4) PS sends Ω_j to TA for requesting a proxy key.
(5) TA with Ω_j and MK outputs the user's attribute private key $ASK[\Omega_j]$ and inserts Key_U_j into the header of the proxy key chain.
(6) TA takes CT_i and $ASK[\Omega_j]$ as input, achieves a decryption to get Key_O_i and inserts it into Key_Pro_j only if Ω_j fully meets the access policy, otherwise output $\perp$, i.e., this user does not have permission to access the data of O_i. This step is repeated until all of the dataset were visited, as a result, generates a complete proxy key chain Key_Pro_j.
(7) TA sends Key_Pro_j to PS.

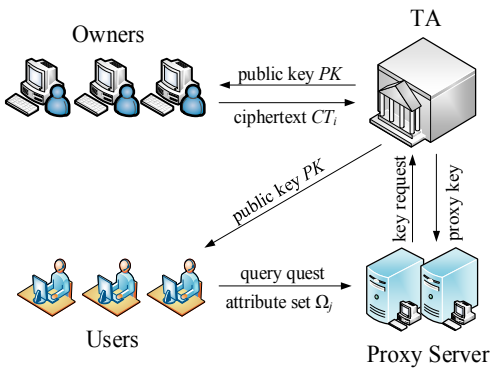

Figure 8. The SVD method for $K = 4$.

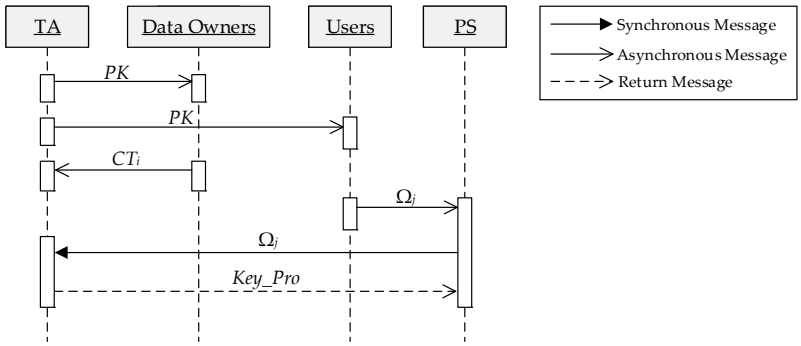

Figure 9. The protocol sequence diagram.

5.4. Result Query

The data user U_j randomly chooses a parameter r_u_j, encrypts the query point Q with Key_U_j and r_u_j, then sends the encrypted query and r_u_j to PS. Next, PS transfers the encrypted query to CS and then destroys it locally (This is a reasonable action, because a person would not keep all secret shares locally for the data security). Through the access control process described in Section 5.3, the proxy gets the key chain Key_Pro_j, i.e., U_j can visit the data in the cloud via a proxy server. Suppose U_j can get a permission for w dataset, then $Key\_Pro_j = \{Key\_U_j, Key\_O_1, \ldots, Key\_O_w\}$. Let $E_{Key\_O,r\_o}(D) = \{E_{Key\_O_1,r\_o_1}(D_1), E_{Key\_O_2,r\_o_2}(D_2), \ldots, E_{Key\_O_w,r\_o_w}(D_w)\}$ denote w corresponding dataset in the cloud, where $E_{Key\_O_i,r\_o_i}(D_i)$ is composed of K_i data items shown in Figure 7.

Firstly, the PS randomly selects a key sk_q and a parameter r_q for a query. The PS and the CS view sk_q and r_q as a normalized key and a normalized parameter for this query, respectively. The encrypted data in the CS were given normalized treatment by DataNorm algorithm. Then the PS and the CS find a block that contains the query point by the SecComp algorithm, i.e., find a block B, making $x_Q > x_{LL}$, $y_Q > y_{LL}$, $x_{UR} > x_Q$ and $y_{UR} > y_Q$. Repeat the above operation over dataset of size w until output an encrypted result point to the U_j. At last U_j decrypts the ciphertext to obtain a nearest neighbor. The process of SNN query is described formally in Algorithm 6.

Algorithm 6. SNN-SAMOMU.

Require: CS has $E_{Key\_O,r\_o}(D)$ and $E_{Key\_U_j,r\_u_j}(Q)$
PS has Key_Pro_j, sk_q and r_q
U_j has Key_U_j and r_u_j.

(1) PS and CS: **FOR** $1 \le i \le w$
  (a) CS get $E_{sk_q,r_q}(D)$ and $E_{sk_q,r_q}(Q)$ by DataNorm algorithm
  (b) get the block B where the nearest neighbor locate by SecComp algorithm
  (c) **WHILE** ($t_k \in B$)
  $E_{sk_q,r_q}(d_k) \leftarrow \text{SecDist}(E_{sk_q,r_q}(Q), E_{sk_q,r_q}(t_k))$
  END WHILE
  (d) $Min_i \leftarrow \text{SecMin}_{K_i}(E_{sk_q,r_q}(d_1), \ldots, E_{sk_q,r_q}(d_{K_i}))$
  (e) get δ where $Min_i == E_{sk_q,r_q}(d_\delta)$
  (f) $Res_i = E_{sk_q,r_q}(t_\delta)$ and $Min_i' = E_{sk_q,r_q}(Min_i)$

(2) PS and CS:
  (a) $Min' \leftarrow \text{SecMin}_w(Min_1', Min_2', \ldots, Min_w')$
  (b) get ξ, where $Min' == Min_\xi'$
  (c) get Res = DataNorm(Res_ξ,{sk_q,r_q},{Key_U_j,r_u_j})

(3) CS:
  (a) Send Res to U_j

(4) U_j:
  (a) $\text{NN} \leftarrow D_{Key\_U_j,r\_u_j}(Res)$

5.5. Optimization

Our scheme runs on the top of encrypted data for the SNN query, whereas it does introduce inefficiency. Now we discuss two strategies to boost the efficiency: offline computation and pipeline execution.

In our protocols, the actual online computation costs with an offline phase can be much less than their costs without an offline phase. For example, consider the DataNorm primitive described in Algorithm 1. During the execution of DataNorm, P_1 has to compute the encrypted value $s = (m_3 g^{a_3} \bmod n)^{-1}$, where m_3, a_3 and r_3 are random numbers in Z_N. However, since these numbers are integers chosen by P_1 at random, the computation of s is independent of any specific factor of DataNorm. That is, P_1 can precompute the value of s during the offline phase, thus reducing its online computation time. In a similar manner, P_1 and P_2 can precompute certain intermediate values in the protocols.

We are able to further reduce the online execution time by adopting the technique of pipeline execution. Take the execution of SecMin$_k$ for instance, P_1 and P_2 would like to process SecComp(d_1, d_2) and SecComp(d_3, d_4). Here the execution of SecComp(d_3, d_4) does not have to wait for the end of SecComp(d_1, d_2). Instead, they can be executed synchronously. We expect that we could further save at least one-third of the online execution time in the long run when we have a lot of SecComp operations to perform. Likewise, we could pipeline the SNN-SAMOMU protocol to save much time.

6. Security Analysis and Performance Evaluation

In this section, we analyze security properties of the proposed scheme, and show that it achieves the defined security design goals. We then provide the performance evaluation on our scheme.

6.1. Security Analysis

6.1.1. Data Confidentiality

In our method, Owners segment the data set through SVD algorithm [15] and encrypt their own data and index by the encryption scheme described in Section 4.1. The data confidentiality in the above process is ensured by the following theorems:

Theorem 1 ([15]). *If E is a secure encryption scheme in a standard security model M, then SVD method is as security as E in the same model M with respect to a single query. For the details of the proof, refer to [15].*

Theorem 2 ([31]). *The encryption scheme described in Section 4.1 can be against chosen plaintext attack (CPA) threat. For the details of the proof, refer to [31].*

Now we analyze that our method can resist adversary *Adv* which achieve data confidentiality. If *Adv* eavesdrop the transmission link between Owners and CS, the encrypted values $E_{Key\_O_i, r\_o_i}(D_i)$ are got by *Adv*. Moreover, all the intermediate values transmitted between PS and CS may also be eavesdroped by *Adv*. Because all these data are transmitted in encrypted form and are randomized by the parameters r_o_i or other random numbers involved in the protocols, it is impossible for *Adv* to decrypt the ciphertext and intermediate values without knowing the Owners' keys or parameters.

Next, suppose *Adv* compromises a specific Owner O_z and CS simultaneously, to get all encrypted data stored in CS, O_z's secret key Key_O_z and parameters r_o_z. However, *Adv* cannot recover the plaintext of other Owners except for O_z. Because all Owners encrypted the data with their own secret keys and random parameters. In addition, all the intermediate values in CS are encrypted with sk_q or randomized by r_q during each query. In all, *Adv* cannot know any assistance to decrypt the encrypted data, i.e., the data confidentiality, defined in Section 3.2, was satisfied.

6.1.2. Query Privacy

Here, we analyze that our method can resist adversary *Adv* which achieve query privacy. If *Adv* eavesdrop the transmission from the User U_z, the encrypted query $E_{Key\_U_z, r\_u_z}(Q)$ are got by *Adv*. However, *Adv* cannot recover the query point without knowing the U_z's secret keys Key_U_z and the parameter r_u_z. Next, suppose *Adv* compromises CS, some Users and Owners simultaneously, to get some Users' and Owners' secret keys and parameters and intermediate values during a query. It is also too hard to get any information that can reveal the actual query point. Because all computations are implemented on encrypted data and all the intermediate values that contain the query point are randomized in the query protocol. In conclusion, the query privacy defined in Section 3.2 was satisfied in our method.

6.2. Performance Evaluation

We developed a Java prototype that implements our method (SNN-SAMOMU). More specifically, we make use of: (a) an Alibaba Elastic Compute Service (ECS) instance with quad-core Intel Haswell CPU at 2.50 GHz, 16 GB RAM as cloud server; (b) a desktop with an Intel(R) 3.30 GHz CPU and 16 GB RAM running Windows 7 as proxy server; and (c) a laptop running Windows 7 with 2.80 GHz CPU and 4 GB RAM as client (data user and owner). The maximum communication bandwidth between the cloud server and the proxy server is set to 10 Mbps, while that between the client and the servers is set to 1 Mbps.

To make a comprehensive performance evaluation, our experiments are implemented on three different datasets (as shown in Figure 10): (a) a real-world dataset from California's Points of Interest [32] which contains 104,770 location records; (b) a synthetic dataset following uniform distribution; and (c) a synthetic dataset following standard normal distribution. We test our scheme over these datasets with different scales of data size (from 20,000 to 100,000). At least 30 random NN queries are selected and evaluated with each scale. In addition, we used the Qhull library to find the Voronoi diagram for the dataset D and used SVD method to ensure each rectangular partition has roughly 1000 points. For encryption scheme, we used 1024-bits keys. Table 2 presents the specific parameter settings in our experiment.

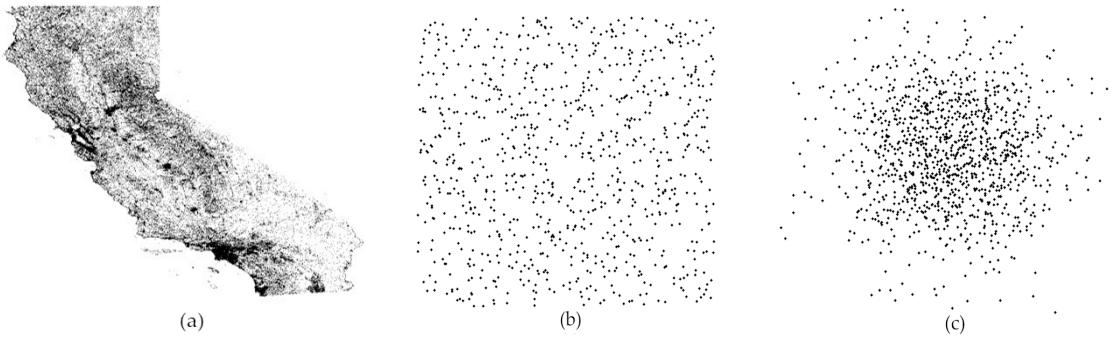

Figure 10. Different datasets: (**a**) real-world; (**b**) uniform distribution; and (**c**) standard normal distribution.

Table 2. Parameter Settings.

Parameter	Values
Maximum communication bandwidth between the cloud server and the proxy server	10 Mbps
Maximum communication bandwidth between the client and the servers	1 Mbps
Size of the dataset $D(\times 10^5)$	0.2, 0.4, 0.6, 0.8, 1.0
Size of each rectangular partition	1000
Size of the keys	1024 bits

The main performance metrics used to evaluate the proposed scheme are data processing time at the data owner, query response time and communication cost at the user. We compare our scheme with two existing schemes, the SVD-SNN method [15] and the VD-1NN method [18].

6.2.1. Data Processing Time at the Data Owner

In the procedure of data pretreatment, there are two major steps for the data owner: performing SVD algorithm and encrypting data. As we can observe from Figure 11, with an increase of the data size, the data processing time increases. It is extremely efficient when the data size is small, but relatively inefficient when the number of records in the dataset reaches 100,000. For instance, it only requires 13.5 s on the real-world dataset (Figure 11a) with 20,000 records, while the data processing time is about 80 s with 100,000 records. However, this is only a one-time cost. Besides, spending more time to build an index in order to optimize query time is the essential methodology.

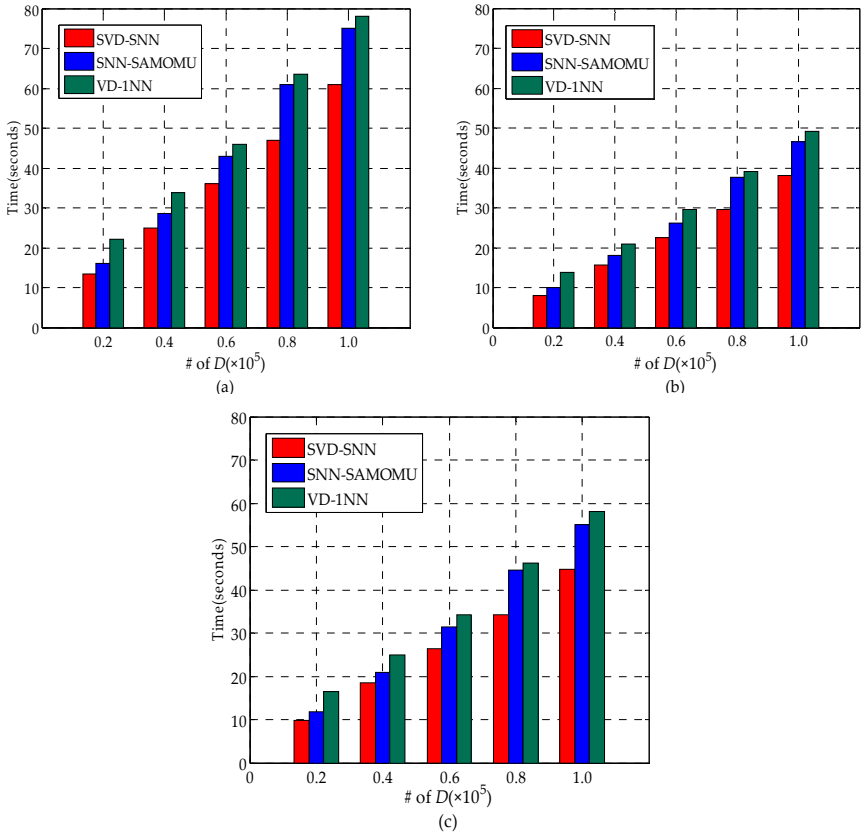

Figure 11. Data Processing Time at the Data Owner: (**a**) real-world; (**b**) uniform distribution; and (**c**) standard normal distribution.

In Figure 11, the data processing time of our scheme is somewhere between SVD-SNN and VD-1NN because the owner in the SVD-SNN encrypts the data with AES, which is an efficient cryptographic primitive, while the owner in VD-1NN has to compute many auxiliary parameters beside of encrypting all the points. Another observation is that these three schemes exhibit the best performance on the uniform dataset (Figure 11c), whereas they show the worst performance on the real-world dataset (Figure 11a). This is because uneven density of the real-world dataset causes SVD algorithm to be highly inefficient.

6.2.2. Query Response Time

The main performance metrics used to evaluate the proposed technique are query response time. This indicator measures the duration from the time the query is issued until the results are received at the end-user. It includes the computation time at the proxy server, the cloud server and the client, as well as the communication time,å which makes up a considerable percentage of total time. Figure 12 shows the query response time for all considered methods under different datasets. As we can see in Figure 12, different distributions have limited effect on query response time for these methods, since all values are treated in a similar way in encrypted form. Furthermore, we can find our scheme is slightly better than others. In order to show the superiority of our method, Figure 13 provides a breakdown of the response time into the server CPU time, the end-user CPU time and the communication time on the real-world dataset. Note that the server CPU time consists of the proxy server CPU time and the cloud server CPU time.

Figure 13a shows the end-user CPU time in our method is significantly less than the SVD-SNN method. It is because the users in the SVD-SNN method have to decrypt the partition contains a lot of candidate points rather than a result point. Figure 13a also shows our method is slightly better than VD-1NN method about the end-user CPU time. Another important observation is that the end-user CPU time in our method remains the same with growth of the database scale because the encryption and decryption operation need to be done only once for each query, regardless of data size. More particularly, the end-user in our method only requires a total of 6 ms on average during the SNN query. These are desirable features for MOMU model, as end-users are lightweight devices with limited computation capabilities.

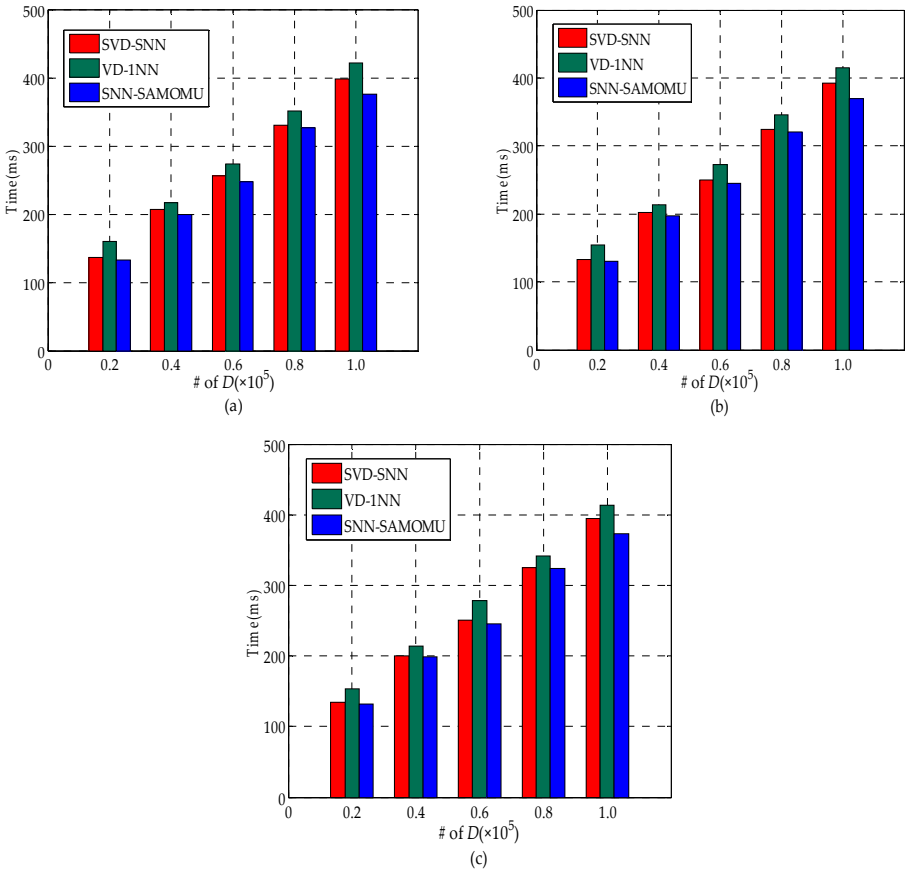

Figure 12. Query response time: (**a**) real-world; (**b**) uniform distribution; and (**c**) standard normal distribution.

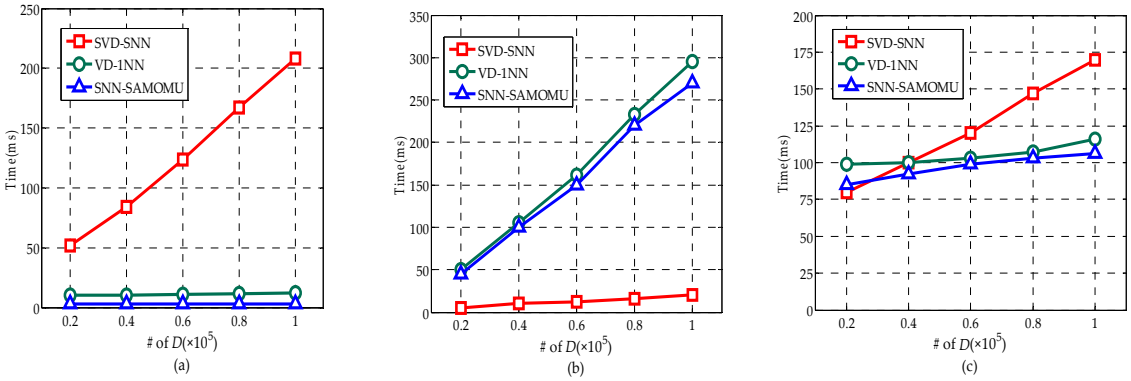

Figure 13. Response time break on the real-world dataset: (**a**) User CPU time; (**b**) Server CPU time; and (**c**) communication time.

In Figure 13b, the lower server CPU time for SVD-SNN is due to the fact that it encrypts the data by AES for increased query efficiency. However, this way decreases the data availability dramatically. The server CPU time in our method is slightly less than the VD-1NN and linearly related to the size of the dataset. Figure 13c shows our method has the best performance for the communication time, which has benefited from the fact that the interactive query time between the proxy and cloud server had the highest proportion of the total time while the interactive time between the user and the server is the most time-consuming in other methods.

6.2.3. Communication Cost at the User

In the experiment, the communication cost is the amount of data transferred between the servers and the user. In Figure 14, it is obvious that the cost in our method is almost negligible while the amount of communication grows with the size of the dataset D in others. This is due to the fact that we use the proxy server to share the hard work for the end-user. However, the users in SVD-SNN are required to receive a large number of indexes and data partition. In VD-1NN, as the result of a mutable order-preserving encryption, the users have to interact with the cloud frequently.

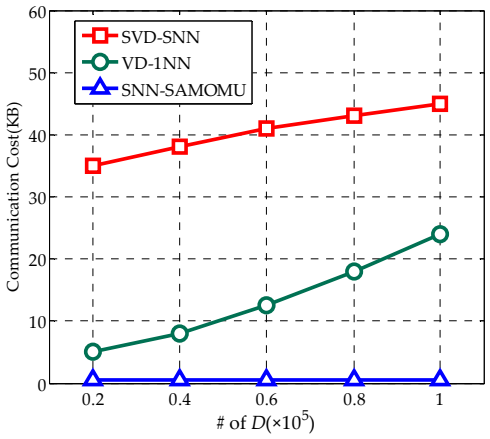

Figure 14. Communication Cost at the user.

7. Conclusions

In this paper, we focus on the secure nearest neighbor (SNN) problem on crowd-sensing location data. The previous SNN techniques generally rely on the Single Owner and Multi Users (SOMU) model, which only contains a single trusted data owner. However, the previous big data system structure has changed because of the crowd-sensing data, i.e., the security and performance requirements

have changed. Given all this, we proposed a SNN query scheme based on the SAMOUMU model, which is constructed by the protocols of secure two-party computation and SVD algorithm. We showed a theoretical analysis that our scheme can protect the data confidentiality and query privacy. Finally, extensive experimental evaluations are presented to show that our scheme is applicable to crowd-sensing data and significantly lower the users' cost. As a future work, we will extend our method to k nearest neighbors and further reduce the server's cost.

Acknowledgments: This work is supported by the National Natural Science Foundation of China (Grant Nos. 61572001 and 61472001).

Author Contributions: Ke Cheng, Liangmin Wang, and Hong Zhong conceived and designed the scheme. Ke Cheng and Liangmin Wang wrote and revised the manuscript. Ke Cheng carried out the experiments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wu, X.; Yang, P.; Tang, S.; Zheng, X. Privacy preserving RSS map generation for a crowdsensing network. *IEEE Wirel. Commun.* **2015**, *22*, 42–48. [CrossRef]
2. Zhang, C.; Subbu, K.P.; Luo, J.; Wu, J. Groping: Geomagnetism and crowdsensing powered indoor navigation. *IEEE Trans. Mob. Comput.* **2015**, *14*, 387–400. [CrossRef]
3. Kotsev, A.; Pantisano, F.; Schade, S.; Jirka, S. Architecture of a service-enabled sensing platform for the environment. *Sensors* **2015**, *15*, 4470–4495. [CrossRef] [PubMed]
4. Bazzi, A.; Zanella, A. Position based routing in crowd sensing vehicular networks. *Ad Hoc Netw.* **2016**, *36*, 409–424. [CrossRef]
5. Wan, J.; Liu, J.; Shao, Z.; Vasilakos, A.V.; Imran, M.; Zhou, K. Mobile crowd sensing for traffic prediction in internet of vehicles. *Sensors* **2016**, *16*, 88. [CrossRef] [PubMed]
6. Wicker, S.B. The loss of location privacy in the cellular age. *Commun. ACM* **2012**, *55*, 60–68. [CrossRef]
7. Narayanan, A.; Shmatikov, V. Robust De-anonymization of Large Sparse Datasets. In Proceedings of the 2008 IEEE Symposium on Security and Privacy (SP 2008), Auckland, New Zealand, 18–22 May 2008; pp. 111–125.
8. Shin, M.; Cornelius, C.; Kapadia, A.; Triandopoulos, N.; Kotz, D. Location privacy for mobile crowd sensing through population mapping. *Sensors* **2014**, *15*, 15285–15310. [CrossRef] [PubMed]
9. Rula, J.P.; Bustamante, F.E. Crowdsensing under (Soft) Control. In Proceedings of the 2015 IEEE International Conference on Computer Communications (INFOCOM'15), Hong Kong, China, 26 April–1 May 2015; pp. 2236–2244.
10. Chen, F.; Zhang, C.; Wang, F.; Liu, J. Crowdsourced live streaming over the cloud. In Proceedings of the 2015 IEEE International Conference on Computer Communications (INFOCOM'15), Hong Kong, China, 26 April–1 May 2015; pp. 2524–2532.
11. Peng, H.; Chen, H.; Zhang, X.Y.; Fan, Y.J.; Li, C.P.; Li, D.Y. Location privacy preservation in wireless sensor networks. *J. Softw.* **2015**, *26*, 617–639.
12. Hu, H.; Chen, Q.; Xu, J. VERDICT: Privacy-preserving authentication of range queries in location-based services. In Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE'2013), Brisbane, Australia, 8–11 April 2013; pp. 1312–1315.
13. Shao, J.; Lu, R.; Lin, X. FINE: A fine-grained privacy-preserving location-based service framework for mobile devices. In Proceedings of the 2014 IEEE International Conference on Computer Communications (INFOCOM'14), Toronto, ON, Canada, 27 April–2 May 2014; pp. 244–252.
14. Peng, H.; Chen, H.; Zhang, X.Y.; Zeng, J.R.; Yun-Cheng, W.U.; Wang, S. Privacy-Preserving k-NN Query Protocol for Two-tiered Wireless Sensor Networks. *Chin. J. Comput.* **2015**, *38*, 872–892.
15. Yao, B.; Li, F.; Xiao, X. Secure nearest neighbor revisited. In Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE'2013), Brisbane, Australia, 8–11 April 2013; pp. 733–744.
16. Wong, W.K.; Cheung, W.L.; Kao, B.; Mamoulis, N. Secure kNN computation on encrypted databases. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD'2009), Providence, RI, USA, 29 June–2 July 2009; pp. 139–152.

17. Hu, H.; Xu, J.; Ren, C.; Choi, B. Processing private queries over untrusted data cloud through privacy homomorphism. In Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE'2011), Hannover, Germany, 11–16 April 2011; pp. 601–612.
18. Choi, S.; Ghinita, G.; Lim, H.S.; Bertino, E. Secure kNN Query Processing in Untrusted Cloud Environments. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 2818–2831. [CrossRef]
19. Elmehdwi, Y.; Samanthula, B.K.; Jiang, W. Secure k-nearest neighbor query over encrypted data in outsourced environments. In Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE'2014), Chicago, IL, USA, 31 March–4 April 2014; pp. 664–675.
20. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the 1999 European Cryptology Conference (EUROCRYPT'99), 2–6 May 1999; Springer: Berlin/Heidelberg, Germany; pp. 223–238.
21. Kido, H.; Yanagisawa, Y.; Satoh, T. Protection of location privacy using dummies for location-based services. In Proceedings of the 21th IEEE International Conference on Data Engineering (ICDE'2005), Tokyo, Japan, 5–8 April 2005; p. 1248.
22. Palanisamy, B.; Liu, L. Mobimix: Protecting location privacy with mix-zones over road networks. In Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE'2011), Hannover, Germany, 11–16 April 2011; pp. 494–505.
23. Gao, S.; Ma, J.F.; Yao, Q.S.; Sun, C. Towards cooperation location privacy-preserving group nearest neighbor queries in LBS. *J. Commun.* **2015**, *36*, 142–150.
24. Chor, B.; Goldreich, O.; Kushilevitz, E.; Sudan, M. Private information retrieval. *J. ACM* **1998**, *45*, 965–981. [CrossRef]
25. Papadopoulos, S.; Bakiras, S.; Papadias, D. Nearest neighbor search with strong location privacy. In Proceedings of the 36th International Conference on Very Large Data Bases (VLDB'2010), Singapore, 13–17 September 2010; pp. 619–629.
26. Yi, X.; Paulet, R.; Bertino, E. Practical k nearest neighbor queries with location privacy. In Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE'2014), Chicago, IL, USA, 31 March–4 April 2014; pp. 640–651.
27. Bugiel, S.; Nürnberger, S.; Sadeghi, A.R.; Schneider, T. Twin clouds: An architecture for secure cloud computing. In *Workshop on Cryptography and Security in Clouds (WCSC'2011)*; ECRYPT II: Zurich, Switzerland, 2011.
28. Liu, X.; Deng, R.; Choo, K.K.R.; Weng, J. An Efficient Privacy-Preserving Outsourced Calculation Toolkits with Multiple Keys. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*. [CrossRef]
29. Wang, B.; Li, M.; Chow, S.S.M.; Li, H. A tale of two clouds: Computing on data encrypted under multiple keys. In Proceedings of the 2014 IEEE Communications and Network Security (CNS'14), San Francisco, CA, USA, 29–31 October 2014; pp. 337–345.
30. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-policy attribute-based encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy(S&P'07), Oakland, CA, USA, 20–23 May 2007; pp. 321–334.
31. Wong, W.K.; Kao, B.; Cheung, D.W.L.; Li, R.; Yiu, S.M. Secure query processing with data interoperability in a cloud database environment. In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD'2014), Snowbird, UT, USA, 22–27 June 2014; pp. 1395–1406.
32. Li, F.; Cheng, D.; Hadjieleftheriou, M.; Kollios, G.; Teng, S.H. On Trip Planning Queries in Spatial Databases. In Proceedings of the 9th International Symposium on Spatial and Temporal Databases (SSTD 2005), Angra dos Reis, Brazil, 22–24 August 2005; pp. 273–290.