Towards Multi-User, Secure, and Verifiable kNN Query in Cloud Database

Ningning Cui, Kang Qian, Taotao Cai, Jianxin Li, *Member, IEEE*, Xiaochun Yang, *Member, IEEE*, Jie Cui, *Senior Member, IEEE*, and Hong Zhong

***Abstract*—With the boom in cloud computing, data outsourcing in location-based services is proliferating and has attracted increasing interest from research communities and commercial applications. Nevertheless, since the cloud server is probably both untrusted and malicious, concerns about data security and result integrity have become on the rise sharply. In addition, in the single-user situation assumed by most existing works, query users can capture query content from each other even though the queries are encrypted, which may incur the leakage of query privacy. Unfortunately, there exists little work that can commendably assure data security and result integrity in the multi-user setting. To this end, in this article, we study the problem of multi-user, secure, and verifiable *k* nearest neighbor query (MSV *k* NN). To support MSV *k* NN, we first propose a novel unified structure, called verifiable and secure index (VSI). Based on this, we devise a series of secure protocols to facilitate query processing and develop a compact verification strategy. Given an MSV *k* NN query, our proposed solution can not merely answer the query efficiently while can guarantee: 1) preserving *data privacy*, *query privacy*, *result privacy*, and *access patterns privacy*; 2) authenticating the *correctness* and *completeness* of the results; 3) supporting *multi-user* with different keys. Finally, the formal security analysis and complexity analysis are theoretically proven and the performance and feasibility of our proposed approach are empirically evaluated and demonstrated.**

***Index Terms*—Data outsourcing, result verification, privacy-preserving, *k*NN query, multiple users.**

I. Introduction

A. Motivation and Background

In recent years, driven by the prosperity of cloud computing, it enhances the power of scalable storage and tremendous computation notably. Intuitively, to support efficient location-based services (LBSs), it is a promising choice for data owners to outsource their data to the cloud server, such as *Amazon* and *Google App* engine. However, under this scenario, the concerns of data security and result integrity are the major brunt. On one hand, the cloud server is untrusted and it may capture or infer sensitive information (e.g., the data in a dataset with name, age, and address or the data in query with location, preferences, and behaviors); on the other hand, the cloud may be compromised or malicious [1] in practice rather than be semi-honest [2]. For instance, the cloud server only implements less computation or returns the tampered results to the users for financial incentives, where the user is unaware of the incorrect results and cannot recognize these. In addition, most existing works are assumed a single-user setting of databases, in which all users share the same key for computability on encrypted data from multiple users. This assumption does have manifest flaws. On one hand, the encrypted database may totally be broken once the unique key is leaked from any compromised user; on the other hand, the query content can be captured by each other. Hence, it is of great urgency to preserve data confidentiality while guaranteeing the result integrity during the query processing in the multi-user environment.

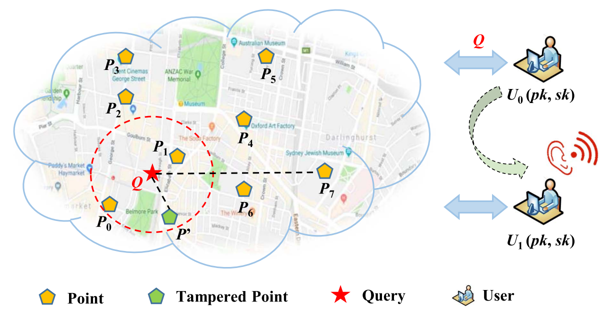


Fig. 1. Sample of the kNN query ($k = 2$).

In this article, we study the problem of multi-user, secure, and verifiable k nearest neighbor query (**MSV** k **NN**), which is prevalent in location-based services. Fig. 1 shows an example of a kNN query $\mathcal{Q}$ of user $\mathcal{U}_0$ with a spatial database $\mathcal{D} = \{\mathcal{P}_0, \ldots, \mathcal{P}_7\}$ and $k = 2$. Here, each location point represents a hotel. We can see that the real 2NN for $\mathcal{Q}$ should be $\{\mathcal{P}_1, \mathcal{P}_0\}$. However, without the integrity verification, the cloud server may return the other two hotels $\{\mathcal{P}', \mathcal{P}_7\}$ for its intended business purpose (e.g., $\mathcal{P}'$ is an unregistered hotel and $\mathcal{P}_7$ is

Manuscript received 26 August 2022; revised 8 December 2022; accepted 14 January 2023. Date of publication 20 January 2023; date of current version 8 August 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 62011530046, U1936220, in part by Industry-university-research Innovation Fund for Chinese Universities under Grant 2020ITA03009, in part by ARC Linkage Project under Grant LP180100750, in part by Ten Thousand Talent Program under Grant ZX20200035, and in part by Excellent Youth Foundation of Anhui Scientific Committee under Grant 2108085J31. Recommended for acceptance by T. Weninger. *(Corresponding author: Jianxin Li.)*

Ningning Cui, Kang Qian, Jie Cui, and Hong Zhong are with the School of Computer Science and Technology, AnHui University, Hefei 230601, China (e-mail: willber1988@163.com; e20301178@stu.ahu.edu.cn; cuijie@mail.ustc.edu.cn; zhongh@ahu.edu.cn).

Taotao Cai is with the School of Computing, Macquarie University, Sydney, NSW 2122, Australia (e-mail: taotao.cai@mq.edu.au).

Jianxin Li is with the School of Information Technology, Deakin University, Geelong, VIC 3220, Australia (e-mail: jianxin.li@deakin.edu.au).

Xiaochun Yang is with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: yangxc@mail.neu.edu.cn).

Digital Object Identifier 10.1109/TKDE.2023.3237879

TABLE I
SUMMARY OF EXISTING kNN QUERY WORKS

Method	Data Privacy	Query Privacy	Result Privacy	Access patterns	Accurate	Verifiable	Multi-user
Wong [3]	×	√	√	×	√	×	×
Choi [5]	√	√	√	×	√	×	×
Elmehdwi [10]	√	√	√	√	√	×	×
Kim [11]	√	√	√	√	√	×	×
Yi [21]	√	√	√	×	×	×	×
Lei [2]	√	√	√	×	×	×	×
Rong [18]	×	√	√	×	×	√	×
Cui [30]	√	√	√	√	√	√	×
Cheng [33]	√	√	√	×	√	×	√

'√' represents the approach satisfies the condition; '×' represents it fails to satisfy the condition.

a far-off hotel). But the query user cannot recognize the result integrity. Whereupon, our MSV k NN provides a verification mechanism to guarantee the following two aspects [1], [23], [24], [25], [26]: 1) *correctness*, i.e., no tampered point ($\mathcal{P}'$) and 2) *completeness*, i.e., no unreal kNN answer ($\mathcal{P}_7$). Meanwhile, without privacy preservation, sensitive information like dataset $\mathcal{D}$, result $\mathcal{R}$, and query $\mathcal{Q}$ are all exposed to the cloud server. To avoid leaking privacy, our MSV k NN aims to preserve the four widely adopted aspects [10], [11], [28]: 1) *data privacy*, i.e., the content of dataset $\mathcal{D}$, 2) *query privacy*, i.e., the content of the query $\mathcal{Q}$, 3) *result privacy*, i.e., the content of the result $\mathcal{R}$, and 4) *access patterns privacy*, i.e., the positions of the result $\{\mathcal{P}_1, \mathcal{P}_0\}$ in $\mathcal{D}$. In addition, under the single-user setting, $\mathcal{U}_1$ can capture the query content and result of $\mathcal{U}_0$ and decrypt them using the same key with $\mathcal{U}_0$. Therefore, our MSV k NN provides a search scheme for *multiple users* as well.

B. Limitations of Prior Art

Recently, there exist various approaches to tackle secure kNN query problem (e.g., Asymmetric-Scalar-Product-Preserving Encryption (ASPE) [3], [4], Order Preserving Encryption (OPE) [5], [6], Searchable Symmetric Encryption (SSE) [2], and Private Information Retrieval (PIR) [7]). However, all these works cannot guarantee the above requirements completely (e.g., *access patterns privacy*). Especially, as [8], [9] indicated, with some prior knowledge about the dataset, the attackers can do the inference attack and even recover the content of the query with the access patterns information. Therefore, it is essential to preserve the access patterns information during the query processing. According to this fact, [10], [11] study the way to protect data access patterns in kNN query, but the efficiency is not acceptable due to the high cost. Besides, as the above works, the cloud server is assumed as semi-honest that could return the correct results. But in reality, it may be malicious that tampers the query result for some unknown incentives.

Unfortunately, so far, little work has been done for secure and verifiable kNN query. In a recent piece of work [18], the authors propose a probabilistic verifiable framework for privacy-preserving kNN query. However, the key weakness is that they utilize ASPE as encryption scheme which has been proven to be insecure and may reveal data and access patterns privacy to the cloud [19]. Our initial study [30] addresses the secure and verifiable kNN query issue (MV k NN), but the performance is unsatisfactory and this work does not support multi-user setting.

Moreover, for the multi-user situation, there also exists little work to deal with kNN query issue directly. Only a piece of work [33] proposes a scheme for secure kNN query under multiple users, but it cannot support the access pattern privacy protection and result verification. Meanwhile, under multi-user scenario, [31], [34], [36], [37], [38] and [39] have explored keyword search, range query and skyline query, respectively. However, all of them cannot be applied to kNN query directly.

For ease of exhibition, we summarize the above works in Table I. Notice that, no existing work can satisfy the whole conditions and can be applied to our problem directly. Inspired by this, it is imperative to design an efficient scheme to guarantee multi-user, secure, and verifiable kNN query.

C. Challenges and Contributions

In this article, we first formally define the problem of multi-user, secure, and verifiable kNN query (MSV k NN). At present, to address this problem, there still exist two key technical challenges.

1) *How to design a unified index and authentication structure for the secure kNN query while guaranteeing the result integrity in multi-user setting?*

That is, in multi-user setting, the verification processing is executed along with the query processing without leaking any privacy. To this end, under distributed two-trapdoor public key cryptosystem (DT-PKC) [35], we propose a verifiable and secure index structure (*VSI*) based on the Voronoi diagram. The VSI is constituted of two parts: *Encrypted Partitioned Grid* and *Bucket-based Encrypted Voronoi Diagram*. To support performing the search over the secure index, a series of secure protocols are put forward (e.g., secure grid computation protocol and secure cell 'read' protocol).

2) *How to accelerate the efficiency further?*

For a traditional kNN query, '*compute and compare*' operation is a crucial step for kNN query processing. But it is rather difficult to execute over encrypted data and is time-consuming generally. To handle this scenario, we develop a novel and efficient *secure minimum distance* protocol, which combines the functionalities of distance computation and comparison but not sacrifice the privacy requirements. Note that it is at least $\times$**70** and $\times\mathbf{5}$ faster than the state-of-the-art schemes [15] and [30], respectively.

In addition to our initial conference version [30], this work extends our initial study in the following aspects: i.) extending and formulating the problem of multi-user, secure, and verifiable kNN query (MSV k NN); ii.) improving the secure and verifiable index structure to efficiently support MSV k NN query; iii.) proposing a series of secure protocols to answer MSV k NN query; iv.) conducting a more comprehensive performance evaluation, which evaluates the the proposed algorithms.

To summarize, our contributions are four-fold as follows.

- To the best of our knowledge, this is the first effort to investigate the MSV k NN preserving data privacy, query privacy, result privacy, and access patterns privacy while guaranteeing the correctness and completeness of the result in multi-user setting.
- We design a verifiable and secure index (VSI) to support MSV k NN query, which combines the authenticated data structure with secure index together.
- We propose a series of secure protocols to facilitate the operations on VSI, which guarantees the secure and verifiable operations and will not bring too much time overhead.
- We conduct the complexity and security analysis and show the feasibility of our scheme in the experiment.

The rest of this article is organized as follows. Section II presents the problem formulation and the preliminaries. Then, Section III introduces the verifiable secure index and describes the auxiliary secure protocols. Next, Section IV presents the query and verification processing. Section V gives the complexity and security analysis. Section VI gives a comprehensive experimental evaluation. We elaborate the related work in Section VII and followed by a conclusion in Section VIII.

II. Preliminaries And Problem Formulation

In this section, we first introduce the system framework, then describe the security model and finally formalize the problem of MSVkNN. A summary of notations and terminologies used in this article is given in Table II.

A. System Framework

In the cloud environment, to meet the privacy requirements (especially the access patterns privacy), we adopt the framework with two collude-resistant clouds, which is commonly used in the related domains [15], [30], [35], as shown in Fig. 2. In practice, these two clouds can be competitive companies, such as *Google* and *Amazon*, who are highly improbable to conspire with each other.[1,2] The specific implementation is described as follows.

- *Certified Authority:* During system setup, the certified authority assigns a pair of public-private keys (pk_c, sk_c) and a secret key (K) to data owner, assigns a pair of public-private keys (pk_u^i, sk_u^i) and a secret key (K) to each user $\mathcal{U}_i$ and assigns partial strong private key SK_1 and all public keys $\{pk_c, pk_u^i\}$ to cloud C_1 and partial strong private key SK_2 and all public keys $\{pk_c, pk_u^i\}$ to cloud C_2, respectively (step ①).
- *Data Owner:* Data owner builds the index $\mathcal{I}$, encrypts the database and index using pk_c (i.e., encrypted database $\mathcal{D}^*$ and secure index $\mathcal{I}^*$) and outsources them to C_1 (step ② and ③).
- *User:* The user $\mathcal{U}_i$ submits a kNN query request $\mathcal{Q}$ to C_1 in the form of ciphertext (i.e., trapdoor $\mathcal{TD}$ encrypted by pk_u^i), (step ④). After receiving the query result and authentication information from the cloud server, denoted as $\mathcal{R}$ and $\mathcal{VO}$, respectively, the query user calculates the final result and verifies the result further (step ⑥).
- *Cloud Server:* Upon receiving the trapdoor $\mathcal{TD}$, the server C_1 cooperating with C_2 carries out the designed secure protocols over the secure index $\mathcal{I}^*$ and trapdoor $\mathcal{TD}$ (step ⑤), and then returns query result and authentication information to the user.

[1]https://www.pcloud.com/encrypted-cloud-storage.html
[2]https://www.boxcryptor.com/en/provider

TABLE II
Summary of Primary Notations

Notation	Meaning
$\mathcal{D}, \mathcal{D}^*$	A dataset and the encrypted one
$\mathcal{P} = \{p_x, p_y\}$	An object in $\mathcal{D}$
$\mathcal{Q} = \{(q_x, q_y), k\}, \mathcal{TD}$	A kNN query and the trapdoor
$\mathcal{I}, \mathcal{I}^*$	The index and the secure index
$\mathcal{R}, \mathcal{VO}$	The result set and verification object
$E(\mathcal{G})$	The encrypted partitioned grid
m	The granularity of $E(\mathcal{G})$
$E(\mathcal{BV})$	The encrypted bucket-based Voronoi
b	The number of buckets in $E(\mathcal{BV})$
w	The number of lines in one bucket
$\mathcal{H}(\cdot)$	A cryptographic hash function
$E(\cdot)$	An encryption function with pk_c
$E_u(\cdot)$	An encryption function with pk_u
pk_c, sk_c	The public-private keys for DO
pk_u, sk_u	The public-private keys for users
SK, SK_1, SK_2	Strong private key and partial ones
K	The secret key for $\mathcal{H}(\cdot)$
$\lambda_{\mathcal{G}}$	The number of packed points in $E(\mathcal{G})$
$\lambda_{\mathcal{NVC}}$	The number of packed points in $E(\mathcal{BV})$
r, Φ	A random number and a packed one
κ	The length of a random number

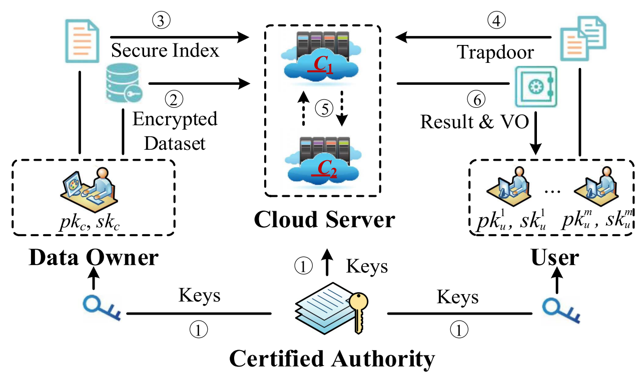


Fig. 2. System framework of the multi-user, secure, and verifiable kNN query in the cloud environment.

B. Problem Definition

In this article, we study the problem of multi-user, secure and verifiable kNN query (MSV k NN). Let a dataset $\mathcal{D} = \{\mathcal{P}_0, \mathcal{P}_2, \ldots, \mathcal{P}_{n-1}\}$, each object $\mathcal{P}$ consists of a two-tuples

$\{p_x, p_y\}$, where p_x and p_y represent the geo-coordinate. In general, given a kNN query $\mathcal{Q} = \{(q_x, q_y), k\}$, where q_x and q_y are the geo-coordinate of query point. Here, MSV k NN aims to find k points owning the minimum distances with the query point while leaking no confidentiality of data and query and ensuring the integrity of the query results in the multi-user setting. The specific definition of MSV k NN is as follows.

Definition 1: (MULTI-USER, SECURE, AND VERIFIABLE kNN QUERY, MSV k NN). *A* MSV k NN *scheme* Π *is constituted of five polynomial algorithms as follows* :

- SETUP$(1^\epsilon) \rightarrow (SK, K, \{pk, sk\})$: is performed by certified authority. It takes a security parameter ϵ as input and outputs SK as the strong private key, K as the signature key, and $\{pk, sk\} = \{(pk_c, sk_c), (pk_u^1, sk_u^1), (pk_u^2, sk_u^2), \ldots, (pk_u^{|\mathcal{U}|}, sk_u^{|\mathcal{U}|})\}$ as the pairs of public-private keys of the data owner and users, respectively.
- INDEXBUILD$(pk_c, \mathcal{D}) \rightarrow (\mathcal{I}^*, \mathcal{D}^*)$: is performed by data owner. It takes public key pk_c and dataset $\mathcal{D}$ as input and outputs the index $\mathcal{I}^*$ and database $\mathcal{D}^*$.
- TRAPGEN$(pk_u^i, \mathcal{Q}) \rightarrow \mathcal{TD}$: is performed by query user $\mathcal{U}_i$. It takes public key pk_u^i and query $\mathcal{Q}$ as input and outputs the encrypted query as trapdoor $\mathcal{TD}$.
- SEARCH$(\mathcal{I}^*, \mathcal{TD}, \mathcal{D}^*) \rightarrow (\mathcal{R}, \mathcal{VO})$: is performed by cloud servers. It takes the secure index $\mathcal{I}^*$, trapdoor $\mathcal{TD}$, and encrypted dataset $\mathcal{D}^*$ as input and outputs query result $\mathcal{R}$ and verification object $\mathcal{VO}$.
- VERIFICATION$(\mathcal{R}, \mathcal{VO}, \mathcal{Q}) \rightarrow$ (*true / false*): is performed by query user. It takes query result $\mathcal{R}$, verification object $\mathcal{VO}$, and query $\mathcal{Q}$ as input and outputs *true* or *false*.

C. Security Model

In this article, we focus on three security threats: (1) the cloud servers are not fully trusted and may intend to tamper the result for the financial incentives; (2) the cloud servers are curious and may attempt to infer confidential information. (3) the users are curious and may attempt to infer query information of other users.

For the first threat, we design a compact and efficient *authenticated data structure* (ADS) to enable the user to verify the query result $\mathcal{R}$. Specifically, the *integrity requirements* involve two aspects:

- *Correctness:* Each returned point $\mathcal{P} \in \mathcal{R}$ is not tampered and is the real point in the original database $\mathcal{P} \in \mathcal{D}$.
- *Completeness:* All returned points are real answers to the kNN query and all non-returned points do not belong to the real answers.

For the other two threats, we design a secure index and a series of novel secure protocols to protect the four kinds of privacy. In detail, the *privacy requirements* are as follows:

- *Data Privacy:* The clouds knows nothing concerning the plaintext of the data in database $\mathcal{D}$;
- *Query Privacy:* The content of the user's query $\mathcal{Q}$ should not be revealed to the clouds and other users.
- *Result Privacy:* The content of the result should not be revealed to anyone else except the query user.
- *Access Patterns Privacy:* The indexs of the points meeting the given kNN query in the database that should be protected from the clouds.

Note that, in our setting, the cloud server C_1 is restricted from compromising with C_2. Moreover, we assume that the collusion among clouds and users should be circumvented during the entire query process.

D. Preliminaries

In this subsection, we introduce some preliminaries for generating secure index and signature.

Cryptographic Hash Function: To verify the correctness of query result, we adopt the one-way cryptographic hash function $\mathcal{H}(\cdot)$ to generate the digital signature $Sig(\cdot)$ of point $\mathcal{P}$.

$$Sig(\mathcal{P}) = \mathcal{H}(\mathcal{P}, K). \tag{1}$$

Notice that, the hash function is collision-resistant, that is for different points $\mathcal{P}_0$ and $\mathcal{P}_1$, it will generate different signatures (e.g., $\mathcal{H}(\mathcal{P}_0) \neq \mathcal{H}(\mathcal{P}_1)$) and cannot recover $\mathcal{P}$ from $\mathcal{H}(\mathcal{P})$. The commonly used hash functions are SHA-1 and SHA-2.

Data Packing (DP)[15]: To reduce the number of times of the encryption and decryption, [15] proposes a way of leveraging message space fully. Intuitively, the message is 1024-bits, and it contains a mass of unoccupied bits. Therefore, [15] packs λ σ-bits integers $x_1, \ldots, x_\lambda$ into one value $< x_1|\ldots|x_\lambda >$, and this value can be computed as follows:

$$< x_1|\ldots|x_\lambda > = \sum_{i=1}^{\lambda} x_i 2^{\sigma(\lambda-i)}. \tag{2}$$

Notice that, we adopt the packing technique to implement all the corresponding protocols. In the aspect of ciphertext, this packed value can be computed as follows:

$$E(x_1|\ldots|x_\lambda) = \Pi_{i=1}^{\lambda} E(x_i)^{2^{\sigma(\lambda-i)}}. \tag{3}$$

Distributed Two Trapdoors Public-Key Cryptosystem: To support multiple users, we adopt the Distributed Two Trapdoors Public-Key Cryptosystem (aka DT-PKC) that is semantically secure [35]. Specifically, DT-PKC mainly contains the following six algorithms:

- WDEC$(sk, E(p)) \rightarrow p$: On input a weak private key sk and ciphertext $E(p)$, it outputs the plaintext p.
- SDEC$(SK, E(p)) \rightarrow p$: On input a strong private key SK and ciphertext $E(p)$, it outputs the plaintext p.
- SKEYS$(SK) \rightarrow (SK_1, SK_2)$: On input a strong private key SK, it outputs two partial strong private keys SK_1, SK_2.
- PSD$(SK_i) \rightarrow D_{SK_i}(p)$: On input a partial strong private key $SK_i, i \in \{1, 2\}$ and ciphertext $E(p)$, it outputs partially decrypted ciphertext $D_{SK_i}(p)$.
- PDC$(D_{SK_1}(p), D_{SK_2}(p)) \rightarrow p$: On input partially decrypted ciphertext $D_{SK_1}(p)$ and $D_{SK_2}(p)$, it outputs the plaintext p.
- CR$(E(p)) \rightarrow E(p)'$: On input the ciphertext $E(p)$ the plaintext p, it outputs another ciphertext $E(p)'$ of the plaintext p.

In addition, DT-PKC has the characteristics of addition homomorphism, given $\forall p_1, p_2 \in \mathbb{Z}_N$:

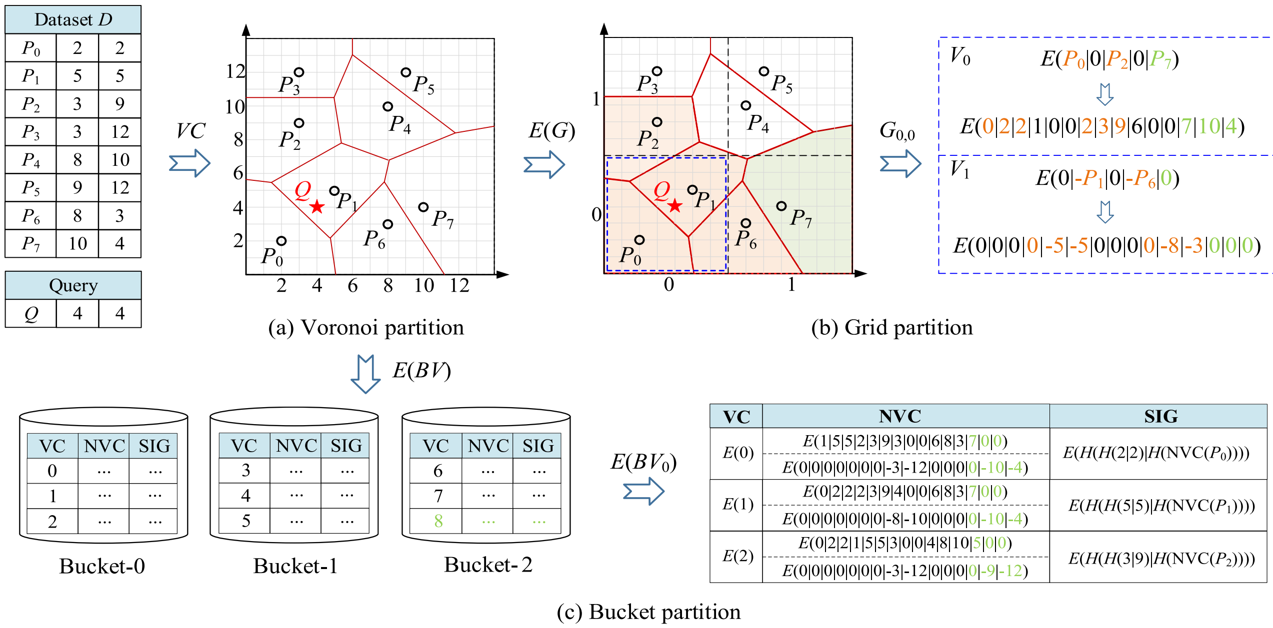


Fig. 3. Example of the verifiable and secure index (*VSI*). In VSI, (a) is the Voronoi partition of dataset $\mathcal{D}$; (b) is the grid partition. In each grid $\mathcal{G}_{i,j}(0 \leq i, j \leq 1)$, it packs the id and coordinates of the covered cells and forms two packed values (i.e., V_0 and V_1). Here, it shows $\mathcal{G}_{0,0} = \{\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_6\}$ for illustration. Since the maximum number of covered cells is 5 (i.e., $\mathcal{G}_{0,1}$), we pad the cell of point $\mathcal{P}_7$ into $\mathcal{G}_{0,0}$ (the points or cell with green are padding records). Moreover, we set the pre-set positions to be $\{1, 3\}$. Therefore, in V_0, the spatial coordinates of points $\mathcal{P}_1$ and $\mathcal{P}_6$ are set to be 0 while in V_1 the ids and spatial coordinates of points $\mathcal{P}_0$, $\mathcal{P}_2$, and $\mathcal{P}_7$ are set to be 0 and the spatial coordinates of $\mathcal{P}_1$ and $\mathcal{P}_6$ are set to negative values; (c) is the bucket partition ($n = 8$, $w = 3$ and $\mathcal{P}_8$ is a padding record). Each bucket consists of one table, where each row represents a cell and contains three columns: i.) *VC* contains the encrypted id of Voronoi cell; ii.) *NVC* contains its adjacent cells of the current Voronoi cell, which is also represented as two packed values following the way of V_0 and V_1; and iii.) *SIG* is the encrypted signature of the current Voronoi cell.

- *Addition:* $E(p_1 + p_2) = E(p_1) \cdot E(p_2)$.
- *Scalar Multiplication:* $E(p_2 \cdot p_1) = E(p_1)^{p_2}$.

III. Our MSVkNN Constructions

In this section, we first introduce our verifiable and secure index structure (*VSI*), which can support verifiable and secure kNN search in the multi-user setting. Then, we propose several well-designed secure protocols to enable private kNN search operations on VSI.

A. Verifiable and Secure Index

To support MSVkNN, we propose a novel data structure, called verifiable and secure index VSI based on the Voronoi diagram, which combines the secure index and authenticated data structure. It mainly consists of two parts: 1). encrypted partitioned grid $E(\mathcal{G})$; 2). bucket-based encrypted Voronoi diagram $E(\mathcal{BV})$.

Encrypted Partitioned Grid $E(\mathcal{G})$. Intuitively, given the dataset $\mathcal{D} = \{\mathcal{P}_0, \ldots, \mathcal{P}_{n-1}\}$, the Voronoi diagram of $\mathcal{P}_i(0 \leq i \leq n-1)$ is a region that the distance of any point lying in this region to $\mathcal{P}_i$ is closer than to any other point $\mathcal{P}_j(i \neq j)$ in $\mathcal{D}$. Hence, the space can be divided into n disjoint convex polygonal regions and each region is called a Voronoi cell $\mathcal{VC}$ belonging to a unique point $\mathcal{P}$. Here, $\mathcal{P}$ is called the generator of this Voronoi cell, as shown in Fig. 3(a). In addition, the generators of $\mathcal{P}'$s neighbor cells are called $\mathcal{NVC}(\mathcal{P})$. It is well known that the Voronoi diagram is always used for fast kNN query due to its following properties.

Property 1: Let $\mathcal{P}$ be the nearest neighbor of the given query point $\mathcal{Q}$. Then, there must exist $\mathcal{Q} \in \mathcal{VC}(\mathcal{P})$.

Property 2: Let $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$ be the top k ($k \geq 2$) nearest neighbors to the given query point $\mathcal{Q}$. Then, there must exist $\mathcal{P}_k \in \{\mathcal{NVC}(\mathcal{P}_1) \cup \mathcal{NVC}(\mathcal{P}_2) \cup \ldots \cup \mathcal{NVC}(\mathcal{P}_{k-1})\}$.

To enable a fast kNN search, an essential step is to check which Voronoi cell the query locates privately. To do this, we propose a grid partition approach over the Voronoi diagram. The main idea is that we first utilize a width Δ to partition the diagram into $m * m$ grids, and for each grid, we record the ids and their spatial coordinates of the overlapped Voronoi cells' generator and form two packed values (e.g., V_0 and V_1 shown in Fig. 3(b)). Specifically, in the packed value V_0, we first choose some points from all covered points according to the pre-set positions (e.g., $\mathcal{P}_1$ and $\mathcal{P}_6$) and then set their spatial coordinates to zero. Further, in V_1, we first set the ids and spatial coordinates of unchosen points to zero and then set the ids and spatial coordinates of chosen points to zero and negative values, respectively. The purpose of this is to protect the relative position relationship between the points and the query when performing secure protocols. In addition, to guarantee each grid containing the same number of the overlapped Voronoi cells, we pad dummy points into grids before generating two packed values (e.g., the green records in the packed value shown in Fig. 3(b)). Then, we pack the ids and coordinates of each generator of the overlapped Voronoi cells in each grid into two packed values V_0 and V_1 and store them in a matrix $\mathcal{G}$. At last, these packing grids are encrypted using the public key of the data owner, denoted as $E(\mathcal{G})$. Note that, to facilitate locating, we also store the encrypted width of a grid $E(\Delta)$ along with the encrypted partitioned grid $E(\mathcal{G})$.

Bucket-Based Encrypted Voronoi Diagram $E(\mathcal{BV})$: Specifically, the encrypted Voronoi diagram $E(\mathcal{BV})$ is used for storing the Voronoi cell, neighbors' Voronoi cells and authenticated information (i.e., signature) of each point $\mathcal{P}$ in $\mathcal{D}$ in the form of ciphertext. When building the bucket-based encrypted Voronoi diagram $E(\mathcal{BV})$, the data owner uniformly partitions the entire table $E(\mathcal{V})$ into buckets, each of which contains w records. Since the last bucket may not be filled fully, to avoid suffering from inference attack based on the number of records (e.g., the clouds can know whether two queries have accessed the identical bucket based on the number of records in bucket), the data owner generates some records randomly and pads them into the last bucket. Note that the padding records will not affect the precision since the padding ids are not in $\mathcal{D}$ and will never be accessed. Recall that the data packing enables multiple records to be encrypted into one ciphertext. As shown in Fig. 3(c), in the first column, *VC* stores the id of each point in the form of ciphertext. In the second column, *NVC* packs the neighbors' VC following the way of generating two packed values (i.e., V_0 and V_1) in $E(\mathcal{G})$. Likewise, to avoid the inference attack, we pad some dummy records into each row of NVC to guarantee the identical size. The dummy records are chosen from other Voronoi cells randomly except the current neighbor cells, which will not affect the query result. In the final column, it is the encrypted signature of each point. According to (1), for a point $\mathcal{P}$, *SIG* contains two parts and can be computed as (4) and (5).

$$\mathcal{H}(\mathcal{NVC}(\mathcal{P})) = \mathcal{H}(\mathcal{H}(\mathcal{P}_{NVC1})|\dots|\mathcal{H}(\mathcal{P}_{NVC\max})) \tag{4}$$

$$Sig(\mathcal{P}) = \mathcal{H}(\mathcal{H}(\mathcal{P})|\mathcal{H}(\mathcal{NVC}(\mathcal{P})) \tag{5}$$

, where max is the maximum number of the neighbor cells.

B. Trapdoor Generation

Given a query $\mathcal{Q} = \{(q_x, q_y), k\}$, before submitting to the cloud server, the user encrypts it using his or her own public key pk_u to safeguard the query privacy.[3] In addition, to facilitate the private search operations on VSI, the trapdoor is constituted of three parts, denoted as

$$\boldsymbol{\mathcal{TD}} = \left\{ \begin{array}{c} (E_u(q_x), E_u(q_y)) \\ (E_u(\underbrace{q_x|q_y|0|0|\dots|q_x|q_y}_{\lambda_{\mathcal{G}}}), E_u(\underbrace{0|0|-q_x|-q_y|\dots|0|0}_{\lambda_{\mathcal{G}}})) \\ (E_u(\underbrace{q_x|q_y|q_x|q_y|\dots|0|0}_{\lambda_{\mathcal{NVC}}}), E_u(\underbrace{0|0|0|0|\dots|-q_x|-q_y}_{\lambda_{\mathcal{NVC}}})) \end{array} \right\} \tag{6}$$

where $\lambda_{\mathcal{G}}$ and $\lambda_{\mathcal{NVC}}$ represent the number of packed points in partitioned grid $E(\mathcal{G})$ and partitioned bucket $E(\mathcal{BV})$, respectively.

C. Secure kNN Protocols

According to VSI, to dispel the security concerns, it should achieve two goals: 1) 'read' the objective grid in $E(\mathcal{G})$ privately and 2) 'read' the objective row in $E(\mathcal{BV})$ securely.

1) Secure Grid Computation Protocol: For the former goal, it mainly has two key points: i.) how to securely obtain the grid coordinate containing the query (i.e., the quotient $E(q_x/\Delta)$ and $E(q_y/\Delta)$ for a given query $\mathcal{Q} = (q_x, q_y)$); ii.) how to blindly 'read' the content of the objective grid (i.e., the content of $\mathcal{G}_{0,0} = \{E(V_0), E(V_1)\}$ in Fig. 3(b)). Hence, we propose a secure division computation protocol and a secure grid computation protocol.

Secure Division Computation (SDC): SDC aims to compute the quotient $E(q_x/\Delta)$ securely. However, the quotient cannot be obtained from the existing protocols directly. Fortunately, we rely on the equality of the equation to construct the following equation:

$$q_x/\Delta = (q_x * r_1 + \Delta * r_1 * r_2)/(\Delta * r_1) - r_2, \tag{7}$$

where $q_x, \Delta \in \mathbb{Z}_N$, and $r_1, r_2 \in \mathbb{Z}_N$ are random numbers. We can see that each part to the right of the equation can be computed securely based on the characteristics of addition homomorphism. Moreover, these random numbers will not have an effect on the quotient but can be applied to obfuscate the real values of q_x and Δ. The overall steps of SDC are shown in Algorithm 1. Based on the property of homomorphic addition, C_1 computes the randomized values $E(q_x')$ and $E(\Delta')$ of q_x and Δ as (7) (Lines 2-3), and partially decrypts them with SK_1 using PSD function to get $D_{SK_1}(q_x')$ and $D_{SK_1}(\Delta')$ (Line 4). Then, C_1 sends them to C_2 (Line 5). Whereupon, C_2 first partially decrypts $E(q_x')$ and $E(\Delta')$ with SK_2 to get $D_{SK_2}(q_x')$ and $D_{SK_2}(\Delta')$ (Line 6). Next, by using PDC function, C_2 obtains the values of q_x' and Δ', computes the quotient h of q_x' and Δ' and sends the encrypted quotient $E(h)$ to C_1 (Lines 7-9). After C_1 received the quotient $E(h)$, based on the equality, C_1 eliminates the random number r_2 from $E(h)$ and obtains the final value of $E(q_x/\Delta)$ (Line 10). Notice that, for any given $q_x \in \mathbb{Z}_N$, 'N - q_x' is equivalent to '-q_x' under $\mathbb{Z}_N$.

Secure Grid Computation (SGC): The goal of SGC is to locate the grid containing the query and 'read' the content from the grid blindly without revealing any information related to the accessed grid to C_1 and C_2. A conventional way is to adopt secure multiplication (*SM*) protocol [10] to do that. However, the time cost is very high that the complexity of the decryption operation is $O(m^2)$. Here, we propose an efficient SGC protocol that reduces the complexity of decryption to $O(m)$ (see details in Section V-A). The process is shown in Algorithm 2.

Initially, for each dimension t, cloud C_1 computes the grid coordinate $\boldsymbol{c}_t$ of the query using SDC protocol with the input $\boldsymbol{\mathcal{TD}}[0][t]$ and $E(\Delta)$ (Line 2) and generates a vector $\boldsymbol{\gamma}_t$ to record the difference between $\boldsymbol{c}_t$ and grid coordinate j ($0 \leq j \leq m -$

[3] It is worth noting that, in the context, we use $E_u(\cdot)$ to represent the encryption function with user's public key pk_u and $E(\cdot)$ to represent the encryption function with data owner's public key pk_c.

Algorithm 1: Secure Division Computation.

Input: C_1 has $E_u(q_x)$, $E(\Delta)$, and SK_1; C_2 has SK_2;
Output: $C_1 \leftarrow E(q_x/\Delta)$;
// Calculation in C_1:
1 Choose two random numbers $r_1, r_2 \in \mathbb{Z}_N$;
2 $E(q_x') \leftarrow E_u(q_x)^{r_1} * E(\Delta)^{r_1 * r_2}$;
3 $E(\Delta') \leftarrow E(\Delta)^{r_1}$;
4 $D_{SK_1}(q_x') \leftarrow$ PSD$(E(q_x'))$ and $D_{SK_1}(\Delta') \leftarrow$ PSD$(E(\Delta'))$;
5 Send $D_{SK_1}(q_x'), D_{SK_1}(\Delta')$ and $E(q_x'), E(\Delta')$ to C_2;
// Calculation in C_2:
6 $D_{SK_2}(q_x') \leftarrow$ PSD$(E(q_x'))$ and $D_{SK_2}(\Delta') \leftarrow$ PSD$(E(\Delta'))$;
7 $q_x' \leftarrow$ PDC$(D_{SK_1}(q_x'), D_{SK_2}(q_x'))$ and $\Delta' \leftarrow$ PDC$(D_{SK_1}(\Delta'), D_{SK_2}(\Delta'))$;
8 $h \leftarrow q_x'/\Delta'$;
9 Send $E(h)$ to C_1;
// Calculation in C_1:
10 $E(q_x/\Delta) \leftarrow E(h) * E(r_2)^{N-1}$;

Algorithm 2: Secure Grid Computation.

Input: C_1 has $E(\mathcal{G})$, $E(\Delta)$ and $\boldsymbol{\mathcal{TD}}[0]$; C_2 has SK_2;
Output: $C_1 \leftarrow E(\mathcal{G}_{ij})$;
// Calculation in C_1:
1 **foreach** *dimension* t **do**
2 $\quad E(\boldsymbol{c}_t) \leftarrow$ SDC$(\boldsymbol{\mathcal{TD}}[0][t], E(\Delta))$;
3 $\quad$ **for** $j = 0$ *to* $m-1$ **do**
4 $\quad\quad E(\boldsymbol{\gamma}_{tj}) \leftarrow E(\boldsymbol{c}_t) * E(j)^{N-1}$;
5 $\quad\quad \boldsymbol{\eta}_{tj} \leftarrow E(\boldsymbol{\gamma}_{tj})^{r_{tj}}$ // r_{tj} is a random number;
6 **for** $i = 0$ *to* $m-1$ **do**
7 $\quad$ **for** $j = 0$ *to* $m-1$ **do**
8 $\quad\quad$ Generate two pairs of $\lambda_{\mathcal{G}}$ random numbers and pack them to Φ_{ij}^0 and Φ_{ij}^1;
9 $\quad\quad E(V_0') \leftarrow E(V_0) * E(\Phi_{ij}^0)$, $E(V_1') \leftarrow E(V_1) * E(\Phi_{ij}^1)$;
10 $\{\boldsymbol{\eta}_0', \boldsymbol{\eta}_1'\} \leftarrow (\pi_1(\boldsymbol{\eta}_0), \pi_2(\boldsymbol{\eta}_1))$ and $\mathbb{G} \leftarrow \pi_2(\pi_1(E(\mathcal{G}')))$;
11 Pack the $\nu_0 \leftarrow \boldsymbol{\eta}_0' + T$ and $\nu_1 \leftarrow \boldsymbol{\eta}_1' + T$;
12 $\nu_0' \leftarrow$ PSD(ν_0) and $\nu_1' \leftarrow$ PSD(ν_1) ;
13 Send ν_0', ν_1', ν_0, ν_1, $\mathbb{G}$ and T to C_2 ;
// Calculation in C_2:
14 $\nu_0'' \leftarrow$ PSD(ν_0) and $\nu_1'' \leftarrow$ PSD(ν_1);
15 $\nu \leftarrow$ PDC(ν_0', ν_0'') and $\nu' \leftarrow$ PDC(ν_1', ν_1'') ;
16 Unpack the $\boldsymbol{\eta} \leftarrow \nu - T$ and $\boldsymbol{\eta}' \leftarrow \nu' - T$;
17 **for** $i = 0$ *to* $m-1$ **do**
18 $\quad$ **for** $j = 0$ *to* $m-1$ **do**
19 $\quad\quad$ **if** $\boldsymbol{\eta}_i == 0$ & $\boldsymbol{\eta}_j' == 0$ **then**
20 $\quad\quad\quad \mathbb{G}' \leftarrow \mathbb{G}_{ij}$;
21 $\quad\quad\quad \boldsymbol{M}_{ij} \leftarrow E(1)$;
22 $\quad\quad$ **else**
23 $\quad\quad\quad \boldsymbol{M}_{ij} \leftarrow E(0)$;
24 Send $\mathbb{G}'$ and $\boldsymbol{M}_{ij}$ to C_1;
// Calculation in C_1:
25 $\widetilde{\boldsymbol{M}} \leftarrow \pi_1^{-1}(\pi_2^{-1}(\boldsymbol{M}))$;
26 $E(\Phi_0) \leftarrow \Pi_{i=0}^{m-1}\Pi_{j=0}^{m-1}(\widetilde{\boldsymbol{M}}_{ij})^{\Phi_{ij}^0}$ and $E(\Phi_1) \leftarrow \Pi_{i=0}^{m-1}\Pi_{j=0}^{m-1}(\widetilde{\boldsymbol{M}}_{ij})^{\Phi_{ij}^1}$;
27 $E(V_0) \leftarrow E(V_0') * E(\Phi_0)^{N-1}$ and $E(V_1) \leftarrow E(V_1') * E(\Phi_1)^{N-1}$;

1) (Line 4). This enables the clouds to check whether the query locates in the grid with $\boldsymbol{\gamma}_{tj} = 0$ for each dimension t.

After that, C_1 obfuscates the difference $\boldsymbol{\gamma}_{tj}$ for $0 \le j \le m-1$ using random numbers r_{tj} to get the vector $\boldsymbol{\eta}_t$ (Line 5), where r_{tj} guarantees that the element in $\boldsymbol{\eta}_t$ is an encryption of either 0 or a random number and preserves the exact value of $\boldsymbol{\gamma}_{tj}$. Note that only the grid containing the query is $E(0)$ in $\boldsymbol{\gamma}$ for each dimension. Then, for each grid in $E(\mathcal{G})$, C_1 obfuscates the encrypted grid $E(\mathcal{G}_{ij}) = \{E(V_0), E(V_1)\}$ using the random numbers Φ_{ij}^0 and Φ_{ij}^1, where Φ_{ij} is packed by $\lambda_{\mathcal{G}}$ random numbers and here λ_G is assumed as the maximum number of packed points in one grid (Lines 6-9). Next, C_1 permutes the obfuscated vectors $\boldsymbol{\eta}_0$ and $\boldsymbol{\eta}_1$ in two dimensions using two random permutation functions π_1 and π_2 respectively, then permutes $E(\mathcal{G}')$ in x-dimension using π_1 and permutes $\pi_1(E(\mathcal{G})')$ using π_2. After that, C_1 packs $\boldsymbol{\eta}_0'$ to ν_0 and $\boldsymbol{\eta}_1'$ to ν_1 respectively, partially decrypts them with SK_1 using PSD function to get ν_0', ν_1' and sends them to C_2 (Lines 10-13). Note that, since the values in $\boldsymbol{\eta}'$ are less than 0 probably, to decrypt them successfully, C_1 adds a threshold T to each value in $\boldsymbol{\eta}'$, where T is no less than each value in $\boldsymbol{\eta}'$, and extends the bit length from σ to σ + 1 to avoid the overflow as (8).

$$E(\boldsymbol{\eta}'_1 + T| \ldots |\boldsymbol{\eta}'_{\lambda_{\mathcal{G}}} + T) = \Pi_{i=1}^{\lambda_{\mathcal{G}}} E(\boldsymbol{\eta}'_i + T)^{2^{(\sigma+1)(\lambda_{\mathcal{G}}-i)}}. \tag{8}$$

Subsequently, upon receiving the values, C_2 first partially decrypts ν_0 and ν_1 with SK_2, obtains the plaintext ν and ν', respectively, and unpacks them to $\boldsymbol{\eta}$ and $\boldsymbol{\eta}'$ (Lines 14-16). Then, C_2 constructs a matrix $\boldsymbol{M}$ to indicate the query location. As mentioned above, only the grid containing the query is $E(0)$ in $\mathbb{G}$, so we can get that if $\boldsymbol{\eta}_i$ and $\boldsymbol{\eta}_j'$ are both 0 (Line 19), the corresponding grid $\mathcal{G}_{ij}$ contains the query. Thereupon, C_2 assigns $\mathbb{G}_{ij}$ to $\mathbb{G}'$ and $E(1)$ to $\boldsymbol{M}_{ij}$; otherwise, C_2 assigns $E(0)$ to $\boldsymbol{M}_{ij}$ (Lines 20-23). After this, C_2 sends $\mathbb{G}'$ and $\boldsymbol{M}$ to C_1.

Finally, since $\mathbb{G}'$ is the obfuscated grid containing the query, it is essential to eliminate the random number from $\mathbb{G}'$. Moreover, only objective gird $\mathcal{G}_{ij}$ is $E(1)$ in $\boldsymbol{M}_{ij}$ and others are $E(0)$. Accordingly, C_1 first computes the inverse permutation of $\boldsymbol{M}$ as $\widetilde{\boldsymbol{M}} = \pi_1^{-1}(\pi_2^{-1}(\boldsymbol{M}))$ and then computes the random number $E(\Phi)$ corresponding to the objective gird $\mathcal{G}_{ij}$ as $E(\Phi_0) = \Pi_{i=0}^{m-1}\Pi_{j=0}^{m-1}(\widetilde{\boldsymbol{M}}_{ij})^{\Phi_{ij}^0}$ and $E(\Phi_1) = \Pi_{i=0}^{m-1}\Pi_{j=0}^{m-1}(\widetilde{\boldsymbol{M}}_{ij})^{\Phi_{ij}^1}$. Once obtaining the random number $E(\Phi)$, C_1 can get the objective grid as $E(V_0) = E(V_0') * E(\Phi_0)^{N-1}$ and $E(V_1) = E(V_1') * E(\Phi_1)^{N-1}$ (Lines 25-27).

Example 1: Table III shows how does the Algorithm 2 work in accordance with Fig. 3. Given the query coordinate $Q = \{4, 4\}$, C_1 intends to 'read' the content of $\mathcal{G}_{00}$ from $\mathcal{G}$

TABLE III
EXAMPLE OF ALGORITHM 2 ($t = 2, m = 2, \sigma = 3, \kappa = 2$)

$\boldsymbol{c}$	$\boldsymbol{\gamma}$	r	$\boldsymbol{\eta}$	$\boldsymbol{\eta}'$	T	ν_0	ν_1	$\boldsymbol{\eta}$	$\boldsymbol{\eta}'$	$\boldsymbol{M}$	$\widetilde{\boldsymbol{M}}$
$\boldsymbol{c}_0$ = 0	γ_{00} = 0	r_{00} = 2	$\boldsymbol{\eta}_{00}$ = 0	$\boldsymbol{\eta}'_{00}$ = -1	7					$\boldsymbol{M}_{00}$ = 0	$\widetilde{\boldsymbol{M}}_{00}$ = 1
	γ_{01} = -1	r_{01} = 1	$\boldsymbol{\eta}_{01}$ = -1	$\boldsymbol{\eta}'_{01}$ = 0	7	6|7 =	4|7 =	$\boldsymbol{\eta}_0$ = -1	$\boldsymbol{\eta}'_0$ = -3	$\boldsymbol{M}_{01}$ = 0	$\widetilde{\boldsymbol{M}}_{01}$ = 0
$\boldsymbol{c}_1$ = 0	γ_{10} = 0	r_{10} = 1	$\boldsymbol{\eta}_{10}$ = 0	$\boldsymbol{\eta}'_{10}$ = -3	7	103	71	$\boldsymbol{\eta}_1$ = 0	$\boldsymbol{\eta}'_1$ = 0	$\boldsymbol{M}_{10}$ = 0	$\widetilde{\boldsymbol{M}}_{10}$ = 0
	γ_{11} = -1	r_{11} = 3	$\boldsymbol{\eta}_{11}$ = -3	$\boldsymbol{\eta}'_{11}$ = 0	7					$\boldsymbol{M}_{11}$ = 1	$\widetilde{\boldsymbol{M}}_{11}$ = 0

securely. During this process, for ease of illustration, we set the grid $m = 2$, data length $\sigma = 3$ and random number length $\kappa = 2$. Besides, we also set two permutation functions $\pi_1 = \{2, 1\}$ and $\pi_2 = \{2, 1\}$, respectively. Due to the limitation of the space, we omit the values like packing random number Φ and the partitioned grid $\mathcal{G}$. Please note that all values in the table are encrypted.

2) Secure Cell Read Protocol: After obtaining the content of the objective grid, it also has two following key points: i.) how to securely obtain the point $E(\mathcal{P})$ and its identifier $E(id)$ from the objective grid, which has the minimum distance with the query $E(\mathcal{Q})$; ii.) how to blindly 'read' the content from encrypted Voronoi diagram $E(\mathcal{BV})$ corresponding to $E(\mathcal{P})$. To do this, we first propose a secure minimum distance protocol and then propose a secure cell read protocol.

Secure Minimum Distance (SMD): As a basic building block protocol, secure minimum computation has been widely studied in the field of secure spatial queries (e.g., kNN query [10], skyline query [28], and similarity query [15]). However, these protocols are not efficient in two aspects: i.) these protocols regard distance computation and minimum computation as two independent protocols; ii.) in each part, they also suffer from a mass of encryption and decryption operations. Therefore, we propose a novel secure minimum distance protocol, which integrates the functionality of distance computation and minimum computation but does not sacrifice privacy. In general, it can calculate the minimum distance from a packed value involving only 7 encryption and 6 decryption operations (see details in Section V-A) while obtaining the corresponding $id_{\min}$ of the minimum, which avoids an extra process of computing $id_{\min}$. Algorithm 3 shows the process of using SMD to obtain the minimum distance in the objective grid.

Algorithm 3: Secure Grid Computation.

Input: C_1 has $E(\mathcal{G}_{ij}), \boldsymbol{\mathcal{TD}}[1], SK_1$; C_2 has SK_2, pk_u;
Output: $C_1 \leftarrow E_u(id_{min}), E_u(p_{x_{min}}), E_u(p_{y_{min}})$;
// Calculation in C_1:
1 Choose $3\lambda_{\mathcal{G}} + 1$ random numbers $r_0 \sim r_{3\lambda_{\mathcal{G}}} \in \mathbb{Z}_N$;
2 Pack $r_1 \sim r_{3\lambda_{\mathcal{G}}}$ into Φ_0 and pack $r_2, r_3, r_5, r_6, \ldots, r_{3\lambda_{\mathcal{G}}-1}, r_{3\lambda_{\mathcal{G}}}$ into Φ_1 ;
3 $\nu_0 \leftarrow ((E(V_0) * E(\Phi_0)) * (E(V_1) * E(\Phi_0)))^{r_0}$;
4 $\nu_1 \leftarrow ((\boldsymbol{\mathcal{TD}}[1][0] * E(\Phi_1)) * (\boldsymbol{\mathcal{TD}}[1][1] * E(\Phi_1)))^{r_0}$;
5 $\nu'_0 \leftarrow \textsc{Psd}(\nu_0)$ and $\nu'_1 \leftarrow \textsc{Psd}(\nu_1)$;
6 Send ν'_0, ν'_1 and ν_0, ν_1 to C_2;
// Calculation in C_2:
7 $\nu''_0 \leftarrow \textsc{Psd}(\nu_0)$ and $\nu''_1 \leftarrow \textsc{Psd}(\nu_1)$;
8 $\nu \leftarrow \textsc{Pdc}(\nu'_0, \nu''_0)$ and $\nu' \leftarrow \textsc{Pdc}(\nu'_1, \nu''_1)$;
9 Unpack ν and ν' to $(\boldsymbol{ID}, \boldsymbol{P})$ and $\boldsymbol{Q}$;
10 **for** $i = 0$ *to* $\lambda_{\mathcal{G}} - 1$ **do**
  $\boldsymbol{d}[i] \leftarrow (\boldsymbol{P}[i].x - \boldsymbol{Q}[i].x)^2 + (\boldsymbol{P}[i].y - \boldsymbol{Q}[i].y)^2$;
  if $\boldsymbol{d}[i] < d_{min}$ **then**
    $d_{min} \leftarrow \boldsymbol{d}[i]$;
    $pos \leftarrow i$;
11 $\boldsymbol{\eta}[pos] \leftarrow E_u(1)$ and $\forall j \neq pos, \boldsymbol{\eta}[j] \leftarrow E_u(0)$;
12 Send $\boldsymbol{\eta}, E_u(\boldsymbol{ID}[pos]), E_u(\boldsymbol{P}[pos].x), E_u(\boldsymbol{P}[pos].y)$ to C_1 ;
// Calculation in C_1:
13 $E_u(r_{id}) \leftarrow \Pi_{i=0}^{\lambda_{\mathcal{G}}-1} \boldsymbol{\eta}[i]^{r_{3*i+1}}$, $E_u(r_x) \leftarrow \Pi_{i=0}^{\lambda_{\mathcal{G}}-1} \boldsymbol{\eta}[i]^{r_{3*i+2}}$ and $E_u(r_y) \leftarrow \Pi_{i=0}^{\lambda_{\mathcal{G}}-1} \boldsymbol{\eta}[i]^{r_{3*i+3}}$;
14 $E_u(id_{min}) \leftarrow E_u(\boldsymbol{ID}[pos])^{r_0^{-1}} * E_u(r_{id})^{N-2}$, $E_u(p_{x_{min}}) \leftarrow E_u(\boldsymbol{P}[pos].x)^{r_0^{-1}} * E_u(r_x)^{N-2}$, $E_u(p_{y_{min}}) \leftarrow E_u(\boldsymbol{P}[pos].y)^{r_0^{-1}} * E_u(r_y)^{N-2}$;

To start with, cloud C_1 first randomly selects $3\lambda_{\mathcal{G}} + 1$ numbers $r_0 \sim r_{3\lambda_{\mathcal{G}}} \in \mathbb{Z}_N$, packs $r_1 \sim r_{3\lambda_{\mathcal{G}}}$ into Φ_0 and packs $r_2, r_3, r_5, r_6, \ldots, r_{3\lambda_{\mathcal{G}}-1}, r_{3\lambda_{\mathcal{G}}}$ into Φ_1 (Lines 1-2). It is worth noting that to guarantee the values in V_1 and $\boldsymbol{\mathcal{TD}}[1][1]$ to be positive, $r_{3*i+2} (0 \leq i \leq \lambda_G - 1)$ is greater than $p_{x_{\max}} \in \mathcal{D}$ and $r_{3*i+3} (0 \leq i \leq \lambda_G - 1)$ is greater than $p_{y_{\max}} \in \mathcal{D}$. Then, C_1 randomizes $E(V_0)$ and $E(V_1)$ using Φ_0 and r_0 to obtain ν_0 and randomizes the trapdoor $\boldsymbol{\mathcal{TD}}[1][0]$ and $\boldsymbol{\mathcal{TD}}[1][1]$ using Φ_1 and r_0 to obtain ν_1, respectively (Lines 3-4). After that, C_1 partially decrypts them with SK_1 using Psd function to get ν'_0 and ν'_1 and sends them to C_2 (Lines 5-6).

Subsequently, upon receiving them from C_1, C_2 first partially decrypts ν_0 and ν_1 with SK_2, obtains the plaintext ν and ν' of ν_0 and ν_1 using Pdc function, respectively and unpacks them into three vectors $\boldsymbol{ID}$ containing randomized identifiers, $\boldsymbol{P}$ containing randomized points, and $\boldsymbol{Q}$ containing randomized query (Lines 7-9). Next, due to the identical random numbers in $\boldsymbol{P}$ and $\boldsymbol{Q}$, C_2 can calculate the squared euclidean distance and obtain the position pos in the packed value corresponding to the minimum distance (Line 10). Further, C_2 generates a vector $\boldsymbol{\eta}$ to indicate the position pos, that is, if the position is pos, the value in $\boldsymbol{\eta}[pos]$ is assigned to $E_u(1)$; otherwise, the values in $\boldsymbol{\eta}$ are assigned to $E_u(0)$ (Line 11). After this, C_2 sends $\boldsymbol{\eta}, E_u(\boldsymbol{ID}[pos]), E_u(\boldsymbol{P}[pos].x), E_u(\boldsymbol{P}[pos].y)$ to C_1 (Line 12). Note that, here, we use an extra vector $\boldsymbol{POS}$ to record the position of the point with minimum distance in V_0 or V_1, which enables the user to compute the final coordinate. For example, if the values of x-coordinate and y-coordinate are extracted from V_1, which means the values of x and y-coordinates are negative.

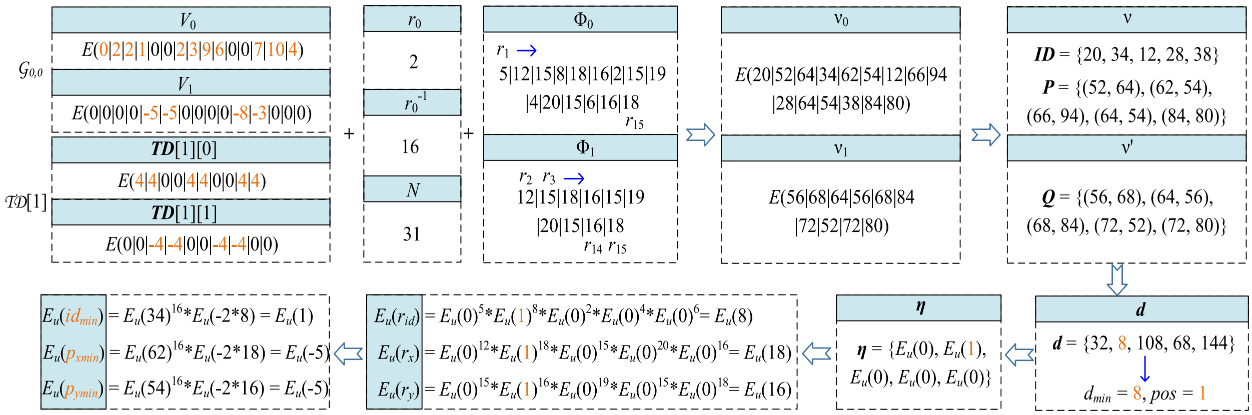


Fig. 4. Example of secure minimum distance (*SMD*) following Algorithm 3. In this example, the objective grid $\mathcal{G}_{00}$ and the query $\mathcal{Q}$ are corresponding to Fig. 3. Here, for ease of illustration, we set the value of N to 31 and omit its effect on encryption and decryption.

At last, since the identifier $E_u(\boldsymbol{ID}[pos])$ and coordinates $E_u(\boldsymbol{P}[pos].x), E_u(\boldsymbol{P}[pos].y)$ are obfuscated, it is essential to eliminate the random numbers from them. Whereupon, based on the vector $\boldsymbol{\eta}$, C_1 computes the random numbers corresponding to the position pos as $E_u(r_{id}) = \Pi_{i=0}^{\lambda_{\mathcal{G}}-1}\boldsymbol{\eta}[i]^{r_{3*i+1}}$, $E_u(r_x) = \Pi_{i=0}^{\lambda_{\mathcal{G}}-1}\boldsymbol{\eta}[i]^{r_{3*i+2}}$ and $E_u(r_y) = \Pi_{i=0}^{\lambda_{\mathcal{G}}-1}\boldsymbol{\eta}[i]^{r_{3*i+3}}$ (Line 13). Further, C_1 obtains the final identifier and coordinates by eliminating the corresponding random numbers as $E_u(id_{\min}) = E_u(\boldsymbol{ID}[pos])^{r_0^{-1}} * E_u(r_{id})^{N-2}$, $E_u(p_{x_{\min}}) = E_u(\boldsymbol{P}[pos].x)^{r_0^{-1}} * E_u(r_x)^{N-2}$ and $E_u(p_{y_{\min}}) = E_u(\boldsymbol{P}[pos].y)^{r_0^{-1}} * E_u(r_y)^{N-2}$, where r_0^{-1} is multiplicative inverse of r_0 under N. (Line 14).

Example 2: As shown in Fig. 4 , it shows the executing process of Algorithm 3. Consider the objective grid is $\mathcal{G}_{00} = \{V_0, V_1\}$ and the query $\mathcal{Q}$ is $(4, 4)$. Here, we mainly list the values of the key variables and omit the partially decrypted values, such as ν_0' and ν_1'. Specifically, C_1 first initiates two packed random numbers $\Phi_0 = 5|12|\ldots|18$ and $\Phi_1 = 12|15|\ldots|18$. Then, it can get the ciphertext ν_0 and ν_1. For example, in ν_0, the first component 20 can be derived from $((0+5)+(0+5))*2$, where two 0s are from the first component of V_0 and V_1, respectively, and 5 is from Φ_0. On the C_2 side, C_2 can obtain the obfuscated sets of the id, point, and query by decrypting and unpacking ν_0 and ν_1. Next, C_2 computes the distance and can get the point with the minimum distance, that is $\boldsymbol{P}[1] = (62, 54)$ and $\boldsymbol{ID}[1] = 34$. After that, on the C_1 side, C_1 first computes the random numbers $E_u(r_{id}), E_u(r_x), E_u(r_y)$ corresponding the position with the minimum (i.e., $pos = 1$) based on $\boldsymbol{\eta}$. Finally, C_1 can further remove the random numbers to get the original id and coordinate. Note that, since the pos in $\boldsymbol{POS}$ is 1, the user can know that the coordinates originating from V_1 are negative and thus can further compute the final result as $p_x = N - p_{x_{\min}} = 31 - 26 = 5$ and $p_y = N - p_{y_{\min}} = 31 - 26 = 5$.

Secure Cell Read (SCR): As mentioned above, the goal of SCR is mainly constituted of two steps: i.) blindly obtaining the bucket containing the given $E(id)$, and ii.) blindly 'read' the cell from the table in the bucket.

Step 1: To obtain the objective bucket, we have the following theorem.

Theorem 1: Assume that the minimum and maximum ids in a bucket are $id_{\min}$ and $id_{\max}$, respectively. Since the point's id in each bucket is ordered, if the given id locates in one bucket, it must have $id - id_{\min} \geq 0$ and $id_{\max} - id \geq 0$; otherwise, it must have $id - id_{\min} < 0$ or $id_{\max} - id < 0$.

As Theorem 1 is apparent, we omit the proof. The specific process of obtaining the objective bucket is as follows.

- On the C_1 side, first, for each bucket $\mathcal{BV}_i$ $(0 \leq i \leq \frac{n}{w})$, C_1 generates two random numbers r_{i0}, r_{i1} and three packing random numbers Φ_{ij}^0, Φ_{ij}^1 and Φ_{ij}^2 $(0 \leq j \leq w-1)$. Then, C_1 computes the two obfuscated differences in the form of the ciphertext, that is $\boldsymbol{\eta}_i = (E(id) * E(id_{\min}^i)^{N-1})^{r_{i0}}$ and $\boldsymbol{\delta}_i = (E(id_{\max}^i) * E(id)^{N-1})^{r_{i1}}$, and likewise C_1 computes the obfuscated bucket as $\Psi_{ij}^0 = \mathbf{VC}_{ij} * \Phi_{ij}^0$, $\Psi_{ij}^1 = \mathbf{NVC}_{ij} * \Phi_{ij}^1$ and $\Psi_{ij}^2 = \mathbf{SIG}_{ij} * \Phi_{ij}^2$. Further, C_1 obtains the permuted $\pi_1(\boldsymbol{\eta}), \pi_1(\boldsymbol{\delta})$, and $\pi_2(\pi_1(\mathcal{BV}))$, where π_1 is used to permute the outer-bucket while π_2 is used to permute the inner-bucket. And C_1 partially decrypts the permuted $\pi_1(\boldsymbol{\eta})$ and $\pi_1(\boldsymbol{\delta})$ using PSD to obtain ν_0 and ν_1. At last, C_1 sends the permuted ν_0, ν_1, $\pi_1(\boldsymbol{\eta}), \pi_1(\boldsymbol{\delta})$, and $\pi_2(\pi_1(\mathcal{BV}))$ to C_2.
- On the C_2 side, C_2 first partially decrypts the $\pi_1(\boldsymbol{\eta}), \pi_1(\boldsymbol{\delta})$ in sequence to obtain ν_0', ν_1'. Then, C_2 invokes PDC function to obtain the clear text $\boldsymbol{\eta}'$ and $\boldsymbol{\delta}'$ of $\boldsymbol{\eta}$ and $\boldsymbol{\delta}$, respectively. Next, C_2 checks if $\boldsymbol{\eta}_i' \geq 0$ and $\boldsymbol{\delta}_i' \geq 0$ $(0 \leq i \leq \frac{n}{w})$, C_2 sets $\mathcal{B} = E(\mathcal{BV}_i)$ and $\boldsymbol{M}_{ij} = E(1)$ for $0 \leq j \leq w-1$; otherwise, $\boldsymbol{M}_{ij} = E(0)$. Subsequently, C_2 returns $\mathcal{B}$ and $\boldsymbol{M}$ to C_1.
- On the C_1 side, C_1 computes the inverse permutation of $\boldsymbol{M}$ as $\widetilde{\boldsymbol{M}} = \pi_1^{-1}(\pi_2^{-1}(\boldsymbol{M}))$. Then, C_1 eliminates the obfuscation from $\mathcal{B}$ as Algorithm 2 (Lines 26-27) and gets the bucket $\mathcal{B}'$ without obfuscation. Now, C_1 has the given id and the matching bucket $\mathcal{B}'$.

Step 2: After obtaining the matching bucket $\mathcal{B}'$, it needs to further obtain the matching cell in this bucket. We can use a similar way to do this and the main idea is that we leverage a vector $\boldsymbol{\eta}$ with length w to indicate the position id in $\mathcal{B}'$ matching with the given id. By utilizing the obfuscation and permutation scheme, C_1 can 'read' a row from $\mathcal{B}'$ blindly following the vector $\boldsymbol{\eta}$.

Discussion: Based on the bucket partition scheme, we merely need to do $\frac{2+4+\cdots+2(\frac{n}{w}+1)}{n}+w$ times decryption operation averagely. Further, we can simplify it as $\frac{n}{w}+w+1$. Hence, it can reach the minimum $2\sqrt{n}+1$ when $w=\sqrt{n}$. In addition, by adopting the data packing technique, we can pack $\boldsymbol{\eta}$ and $\boldsymbol{\delta}$ into one ciphertext to reduce the decryption operation further.

IV. MSV k NN Query Processing

A. Secure kNN Query Processing

Based on *VSI*, we propose a secure and verifiable query processing strategy, which is divided into four steps as follows.

Step 1. Calculating k nearest neighbors: Given the verifiable and secure index *VSI* and the trapdoor $\mathcal{TD}$, C_1 first adopts *SGC* protocol to get the points (i.e., ids and coordinates) in one partitioned grid, where the query point locates in. Then, C_1 computes the minimum distance $d_{\min}$ and its corresponding id between each point in objective grid and the trapdoor $\mathcal{TD}$ using *SMD* protocol. Accordingly, the point with the minimum distance is regarded as the nearest neighbor and is inserted into the result set $\mathcal{R}$. To further get next nearest neighbor (i.e., $2\leq k$), C_1 uses *SCR* protocol to 'read' the row corresponding to the id of the minimum point. After that, C_1 computes the next minimum distance using *SMD* protocol and inserts the other distances along with their ids into the candidate set $\mathcal{C}$. Repeat the above process until $|\mathcal{R}|$ is k. Note that these inserted neighbor points may contain the redundant points which have been the results already or they are identical points, so we should eliminate them from the candidate set to avoid repeated selection or missing results following step 2. Besides, during the query process, we generate the verification object $\mathcal{VO}$ following step 3.

Step 2. Eliminating the redundant points from the candidate set: When $2\leq k$, it involves two conditions:

- *Case 1:* The distances of the points newly inserted to the candidate set are identical with the distances of the points that have been the results already.
- *Case 2:* The distances of the points in candidate set newly inserted to the result set are identical with the distances of the other points in candidate set.

For the former case, since we add all points with minimum distance into result set in each round, based on the property of the Voronoi diagram, there cannot exist an another point owing a more minimum distance than the points in result set. Hence, in such case, we can directly filter the points that have identical distances with the points in result set. For the latter case, we cannot directly check whether these points are same relying on their distances. In this case, we need to check whether the ids of these points are identical. The detailed process is as follows.

• On the C_2 side, for each point, C_2 first generates the same random number r and a vector $\boldsymbol{\eta}$ with length $\lambda_{\mathcal{NVC}}$ to indicate the position i in the row of objective bucket and sets $E_u(1)$ in the matching position i of $\boldsymbol{\eta}$ and sets $E_u(0)$ in other positions. Then, C_2 computes $E_u(\boldsymbol{ID}[i])*E_u(r)$ to get $E_u(\boldsymbol{ID}[i]')$ and sends $E_u(\boldsymbol{ID}[i]')$ and $\boldsymbol{\eta}$ to C_1.

• On the C_1 side, for each point, C_1 first computes the added random number generated in SMD protocol corresponding to the matching position i as $E_u(r_i)=\Pi_{j=0}^{\lambda_{\mathcal{NVC}-1}}\boldsymbol{\eta}[j]^{r_j}$ and eliminates r_i as $E_u(\boldsymbol{ID}[i])=E_u(\boldsymbol{ID}[i]')*E_u(r_i)^{N-1}$. Then, C_1 generates the same random number $r\prime$ for each point and adds it to $E_u(\boldsymbol{ID}[i])$ as $E_u(\boldsymbol{ID}[i]'')=E_u(\boldsymbol{ID}[i])*E_u(r')$. Next, C_1 partially decrypts $E_u(\boldsymbol{ID}[i]'')$ using PSD function with SK_1 to get ν and sends ν along with $E_u(\boldsymbol{ID}[i]'')$ to C_2.

• On the C_2 side, for each point, C_2 first partially decrypts each $E_u(\boldsymbol{ID}[i]'')$ using PSD function with SK_2 to get ν' and obtain the clear text $\boldsymbol{ID}[i]''$ using PDC function with ν and ν'. Then, C_2 eliminates the corresponding random r and checks if the values are identical, this means these points are the same point and need to be filtered; otherwise, they are different points and need to be inserted to the result set.

Step 3. Generating verification object $\mathcal{VO}$: During the search process, we need to generate the verification object simultaneously to support the result authentication. Thanks to our VSI, it can generate the verification object easily. Specifically, this process is involved in SCR protocol. When invoking SCR protocol to obtain an objective row in bucket, we can obtain the information of two columns: NVC and SIG, which is essential to support the result authentication. However, they are encrypted by the public keys of the data owner and cannot be decrypted by users. Hence, before inserting to the $\mathcal{VO}$, it is imperative to transform the public keys of the data owner to the public key of the specific user. The transforming process is as follows.

• On the C_1 side, C_1 first generates two random numbers Φ and r, where Φ is a packed random number. Then, C_1 computes $E(V_0)*E(V_1)^{N-1}*E(\Phi)$ and $E(Sig(\mathcal{P}))*E(r)$, where $\mathcal{P}$ is the point with the current minimum distance, denoted as $E(\eta_0)$ and $E(\eta_1)$. Next, C_1 partially decrypts $E(\eta_0)$ and $E(\eta_1)$ using PSD with SK_1 to get ν_0 and ν_1 and send them with $E(\eta_0)$ and $E(\eta_1)$ to C_2.

• On the C_2 side, C_2 first partially decrypts $E(\eta_0)$ and $E(\eta_1)$ using PSD with SK_2 to get ν_0' and ν_1' and obtains the clear text η_0 and η_1 by PDC with ν_0,ν_1 and ν_0',ν_1', respectively. Then, C_2 encrypts them with pk_u and sends $E_u(\eta_0)$ and $E_u(\eta_1)$ to C_1.

• On the C_1 side, C_1 eliminates the random numbers by computing $E_u(\eta_0)*E_u(\Phi)^{N-1}$ and $E_u(\eta_1)*E_u(r)^{N-1}$ to obtain $E_u(V)$ and $E_u(Sig(\mathcal{P}))$, where $E_u(V)=E_u(V_0)*E_u(V_1)^{N-1}$. At last, C_1 adds $E_u(V)$ and $E_u(Sig(\mathcal{P}))$ of each result point into $\mathcal{VO}$.

Step 4. Returning results and verification object to the user: Based on SMD and SCR protocols, C_1 can obtain the final results encrypted by the public key of the user directly in sequence and does not need to invoke an extra transformed process. Therefore, C_1 puts the final points into result set $\mathcal{R}$ and sends it along with $\mathcal{VO}$ to the user. Moreover, C_2 also sends the vector $\boldsymbol{POS}$ to the user, which is used to indicate the positions of points with minimum distances in V_0 or V_1.

B. Verification Processing

Here, the user utilizes the result set $\mathcal{R}$ and the verification object $\mathcal{VO}$ to authenticate the integrity of the results.

• VERIFYING CORRECTNESS. Recall that the correctness refers to the points in $\mathcal{R}$ all belonging to the original database and not being tampered with by the clouds. To achieve this, we utilize the signature of each point generated by the data owner to guarantee that. Specifically, in $\mathcal{VO}$, it consists of k two-tuples in the form of ciphertext as $\{E_u(V_i), E_u(Sig(\mathcal{P}_i))\}, (1 \le i \le k)$. First, the user invokes the WDEC function with sk_u to decrypt $\mathcal{VO}$, that is $\{V_i, Sig(\mathcal{P}_i)\}$. Since V_i contains all neighbors of point $\mathcal{P}_i$, the user can first unpack V_i and reconstruct the hash value of neighbors of point $\mathcal{P}_i$ as (4). Further, the user can recover the signature directly. That is the user calculates the hash value of point $\mathcal{P}_i$, like $\mathcal{H}(\mathcal{P}_i)$, and then calculates the signature $\widehat{Sig}(\mathcal{P}_i)$ as Eq. 5. At last, if the calculated signature $\widehat{Sig}(\mathcal{P}_i)$ matches the $Sig(\mathcal{P}_i)$ from $\mathcal{VO}$, it means the correctness is verified; vice versa.

• VERIFYING COMPLETENESS. Recall that the completeness refers to the points in $\mathcal{R}$ are the real k nearest neighbors. Based on Property 1 and 2, since the correctness has been verified, the user can verify this aspect using distance comparison and the completeness verification consists of two aspects. On one hand, the distance between point $\mathcal{P}_i$ $(i = 1)$ in $\mathcal{R}$ and $\mathcal{Q}$ must be no more than the distances between $\mathcal{NVC}(\mathcal{P}_1)$ in $\mathcal{VO}$ and $\mathcal{Q}$. If the distance between point $\mathcal{P}_i$ $(i = 1)$ in $\mathcal{R}$ and $\mathcal{Q}$ is less than the distances between $\mathcal{NVC}(\mathcal{P}_1)$ in $\mathcal{VO}$ and $\mathcal{Q}$, the point $\mathcal{P}_1$ is demonstrated to be the nearest neighbor. On the other hand, for the point $\mathcal{P}_i$ $(2 \le i \le k)$ in $\mathcal{R}$, the distance d_i between $\mathcal{P}_i$ and $\mathcal{Q}$ is less than the distances between the points in $\mathcal{NVC}(\mathcal{P}_1) \cup \ldots \cup \mathcal{NVC}(\mathcal{P}_{i-1}) - \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_i$ and $\mathcal{Q}$. To achieve this, the user can first compute the distances of points $\mathcal{P}_i$ $(2 \le i \le k)$ in $\mathcal{R}$. Then, based on the identifier, the user can easily eliminate the result points $\{\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_i\}$ from $\mathcal{NVC}(\mathcal{P}_1) \cup \ldots \cup \mathcal{NVC}(\mathcal{P}_{i-1})$ and further compute the distances between the remained points in $\mathcal{NVC}(\mathcal{P}_1) \cup \ldots \cup \mathcal{NVC}(\mathcal{P}_{i-1})$ and $\mathcal{Q}$. If it is achieved for each point $\mathcal{P}_i$ $(2 \le i \le k)$, the point $\mathcal{P}_i$ is demonstrated to be the ith nearest neighbor. To sum up, if the above two aspects are satisfied, we say that the completeness is verified.

Example 3: As shown in Fig. 3, given the query $\mathcal{Q} = (4, 4)$ and $k = 1$, its NN is $\mathcal{R} = \{\mathcal{P}_1 = (5, 5)\}$ and the verification object is $\mathcal{VO} = \{\mathcal{P}_0|\mathcal{P}_2|\mathcal{P}_4|\mathcal{P}_6|\mathcal{P}_7, Sig(\mathcal{P}_1)\}$, which has been decrypted by the user with sk_u. For the correctness verification, the user first computes the hash value of $\mathcal{P}_1$, that is $\mathcal{H}(\mathcal{P}_1) = \mathcal{H}(5|5)$. Then, the user computes the hash value of its neighbors, that is $\mathcal{H}(\mathcal{NVC}(\mathcal{P}_1)) = \mathcal{H}(\mathcal{H}(\mathcal{P}_0)|\mathcal{H}(\mathcal{P}_2)|\mathcal{H}(\mathcal{P}_4)|\mathcal{H}(\mathcal{P}_6)|\mathcal{H}(\mathcal{P}_7))$. Next, the user recomputes the signature of $\mathcal{P}_1$, that is $\widehat{Sig}(\mathcal{P}_1) = \mathcal{H}(\mathcal{H}(\mathcal{P}_1)|\mathcal{H}(\mathcal{NVC}(\mathcal{P}_1)))$. Finally, the user checks if $\widehat{Sig}(\mathcal{P}_1)$ matches $Sig(\mathcal{P}_1)$, the correctness is satisfied; for the completeness verification, the user first computes the squared distance $d(\mathcal{P}_1, \mathcal{Q}) = 2$, and then computes the squared distances of its neighbors, $d(\mathcal{P}_0, \mathcal{Q}_1) = 8$, $d(\mathcal{P}_2, \mathcal{Q}_2) = 26$, $d(\mathcal{P}_4, \mathcal{Q}_3) = 52$, $d(\mathcal{P}_6, \mathcal{Q}_4) = 17$, and $d(\mathcal{P}_7, \mathcal{Q}_5) = 36$. Since the distance $d(\mathcal{P}_1, \mathcal{Q}) = 2$ is the minimum, it means the query $\mathcal{Q}$ locates in Voronoi cell of point $\mathcal{P}_1$ (i.e., $\mathcal{VC}(\mathcal{P}_1)$). Therefore, the completeness is authenticated.

V. ANALYSIS

A. Complexity Analysis

1) Computation Complexity: Table IV provides the computational complexity of existing approaches and our protocols in secure kNN computation. The comprehensive analysis is exhibited as follows:

• In SDC protocol (Algorithm 1), it requires 2 encryptions in line 9 and line 10 and requires 6 decryptions in line 4, line 6, and line 7.

• In SGC protocol (Algorithm 2), it executes 2 times SDC protocol that leads to 4 encryptions and 12 decryptions in line 2. In addition, it requires $2m^2 + m + 6$ encryptions in lines 4, 9, 21, and 23 and involves $6\lceil m/\lambda \rceil$ decryptions in lines 12, 14, and 15. Note that, $\lceil m/\lambda \rceil$ represents the quantity of the packed values in partitioned grids.

• In SMD protocol (Algorithm 3), it requires 7 encryptions in lines 3, 4, 11, and 12 and requires 6 decryptions in lines 5, 7, and 8. Considering the SMD_n executes the secure minimum distance computation of n points, hence with data packing, this leads to $7\lceil n/\lambda \rceil$ times encryptions and $6\lceil n/\lambda \rceil$ times decryptions, where $\lceil n/\lambda \rceil$ represents the number of the packed values of n points.

• In SCR protocol, in the first step, it requires $3w$ encryptions in each bucket and leads to $\lceil n/w \rceil 3w + 3w + 2$ encryptions in total and $6\lceil n/w\lambda \rceil$ decryptions, where $\lceil n/w \rceil$ represents the quantity of the partitioned buckets. In second step, it involves $3w + 2$ encryptions and $3\lceil w/\lambda \rceil$ decryptions, where $\lceil w/\lambda \rceil$ represents the number of the packed values of indicators (i.e., 0 or 1).

To sum up, in MSV k NN, it mainly involves 1 time SGC protocol and k times SCR protocol. Note that, to minimize the decryption time (details see discussion in Section III-C), we set $w = \sqrt{n}$ and $m^2 \ll n$. Hence, the complexity of MSV k NN is $O(kn)$ encryption and $O(k\sqrt{n}/\lambda)$ decryption.

2) Communication Complexity: In addition, Table V provides the communication complexity of existing and our approaches in secure kNN computation. Here, the communication cost involves the interactions between C_1 and C_2. Specifically, in MSV k NN, the secure grid computation SGC and secure cell read SCR occupy the main communication cost.

• In SGC protocol, there exist two intersections between C_1 and C_2. For the first round, the cost is $O(2m^2C)$; for the second round, the cost is $O(m^2C)$.

• In SCR protocol, there exist two steps: in the first step, there exist two intersections between C_1 and C_2. For the first round, the cost is $O((4n + 2b)C)$; for the second round, the cost is $O((n + 4w)C)$; in the second step, there also exist two intersections between C_1 and C_2. For the first round, the cost is $O(6wC)$; for the second round, the cost is $O(4wC)$.

Overall, in MSV k NN, the communication cost is $O((3m^2 + k(14w + 2b + 5n))C)$.

B) Security Analysis

To analyze the security, we adopt the formal definition of multi-party computation introduced in [15], [28], following the

TABLE IV
COMPUTATIONAL COMPLEXITY OF EXISTING APPROACHES AND OURS ($\lambda = \lfloor K/(\sigma+\kappa+1)\rfloor, t_1+t_2 = log_2 n$)

Protocol	Approach	Computation Cost		
		Encryption	Decryption	non-XOR gates
Secure Minimum Computation	SMIN$_n$ [10] + DPSSED [15]	$(14\sigma+5)n+2\lceil n/\lambda\rceil$	$7\sigma(n-1)+2\lceil n/\lambda\rceil$	0
	SMS$_n$ [15] + DPSSED [15]	$6n+4\lceil n/\lambda\rceil$	$4\lceil n/\lambda\rceil$	$(3\sigma+3\kappa+1)(n-1)$
	SMC$_n$ [30] + DPSSED [15]	$6n+2\lceil n/\lambda\rceil$	$2\lceil n/\lambda\rceil(2-(1/2)^{t_1})+t_2$	0
	SMD$_n$	$7\lceil n/\lambda\rceil$	$6\lceil n/\lambda\rceil$	0
Secure Division Computation	SDC [30]	2	2	0
	SDC	2	6	0
Secure Grid Computation	SGC [30]	m^2+m+4	$2\lceil m/\lambda\rceil+4$	0
	SGC	$2m^2+m+6$	$6\lceil m/\lambda\rceil+12$	0
Secure Cell Read	SCR [30]	$\lceil n/w\rceil 5w+5w+4$	$2\lceil n/w\lambda\rceil+\lceil w/\lambda\rceil$	0
	SCR	$\lceil n/w\rceil 3w+3w+4$	$6\lceil n/w\lambda\rceil+3\lceil w/\lambda\rceil$	0

TABLE V
COMMUNICATION COMPLEXITY OF EXISTING AND OUR APPROACHES (C REPRESENTS THE SIZE OF A CIPHERTEXT; l REPRESENTS THE HEIGHT OF THE KD-TREE; t REPRESENTS THE NUMBER OF POINTS CONTAINED IN THE LEAF NODE OF THE KD-TREE; b REPRESENTS THE NUMBER OF BUCKETS; w REPRESENTS THE NUMBERS OF LINES IN ONE BUCKET; m REPRESENTS THE GRANULARITY OF THE PARTITIONED GRID)

Protocol	Approach	Communication Cost
Secure kNN Computation	Sec$k$NN [33]	$(7*2^l+12t+14tk)C$
	SVkNN [30]	$(2m^2+k(12w+2b+6n))C$
	MSVkNN	$(3m^2+k(14w+2b+5n))C$

framework of simulation paradigm [17]. The main idea is as follows.

Theorem 2. Composition Theorem [17]: Given a protocol Ω consists of some sub-protocols, if all the sub-protocols are secure and all the intermediate results are random or pseudo-random, we say the protocol Ω is secure.

Intuitive, based on the simulation paradigm, it requires that the view of each party participating in a protocol can be simulated relying on its input and output, which implies that the parties can capture nothing from the protocol. In other words, the simulated view of each sub-protocol is computationally indistinguishable from the actual execution view. For the ease of presentation, we formally demonstrate the SMD$_n$ protocol for illustration and other protocols can be demonstrated in the same way.

Theorem 3: The SMD$_n$ protocol is secure for any probability polynomial time adversaries $\mathcal{A}$, if there exists a simulator $\mathcal{S}$ such that the probability Pr($\mathsf{Real}^{\mathcal{A}}_{SMD_n}$) - Pr($\mathsf{Sim}^{\mathcal{A}}_{SMD_n}$) is negligible,

$$|\Pr(\mathsf{Real}^{\mathcal{A}}_{SMD_n}) - \Pr(\mathsf{Sim}^{\mathcal{A}}_{SMD_n})| \leq negl(\epsilon).$$

Proof: To demonstrate this, we first define the real view $\mathsf{Real}^{\mathcal{A}}_{SMD_n}$ and simulated view $\mathsf{Sim}^{\mathcal{A}}_{SMD_n}$.

$\mathsf{Real}^{\mathcal{A}}_{SMD_n}$*:* Given dataset $\mathcal{D} = \{E(\mathcal{P}_1), E(\mathcal{P}_2), \ldots, E(\mathcal{P}_n)\}$, and g queries $\{E_u(\mathcal{Q}_1), E_u(\mathcal{Q}_2), \ldots, E_u(\mathcal{Q}_g)\}$, in the protocol, C_1 and C_2 compute the distances, compare the distances and obtain the minimum distance. Specifically, for each query $E(\mathcal{Q}_i)$, C_1 computes the obfuscated packing points and trapdoor with random numbers Φ_0, Φ_1 and r_0 to obtain ν_0 and ν_1, respectively. Then, C_1 computes the partially decrypted results ν'_0 and ν'_1. Next, C_1 obtains the intermediate results $\{\nu_0, \nu_1, \nu'_0, \nu'_1\}$ and sends them to C_2. C_2 computes the partially decrypted results ν''_0 and ν''_1, compares them and sends the intermediate results $\{\boldsymbol{\eta}, E_u(id), E_u(x), E_u(y)\}$ to C_1. At last, C_1 outputs $E_u(\mathcal{P}_s)$ $(1 \leq s \leq g)$ with $E_u(id_{\min})$ for $E_u(\mathcal{Q}_s)$ by the experiment.

$\mathsf{Sim}^{\mathcal{A}}_{SMD_n}$*:* Simulator $\mathcal{S}$ receives $E_u(\mathcal{P}_s)$, and then generates random points $E(\overline{\mathcal{P}}_z)$ for $1 \leq z \leq n-g$ through generating a point $E(\mathcal{P})$ randomly and multiplying a large random number $E(r_z)$. This guarantees that the distance $d(\overline{\mathcal{P}}_z, \mathcal{Q}_s) > d(\mathcal{P}_s, \mathcal{Q}_s)$. After that, $\mathcal{S}$ forms the simulated dataset $\overline{D} = \{E_u(\mathcal{P}_s), E(\overline{\mathcal{P}}_z)\}$. When executing a query $E_u(\overline{\mathcal{Q}}_s)$ ($\overline{\mathcal{Q}}_s = \mathcal{Q}_s$ and $E_u(\overline{\mathcal{Q}}_s) \neq E_u(\mathcal{Q}_s)$) for $1 \leq s \leq g$, the simulator $\mathcal{S}$ runs the SMD$_n$ protocol, and outputs the results by the experiment.

Intuitively, based on the simulator $\mathcal{S}$, no probability polynomial-time (PPT) adversary can distinguish the $\mathsf{Sim}^{\mathcal{A}}_{SMD_n}$ from $\mathsf{Real}^{\mathcal{A}}_{SMD_n}$ since the outputs of them are identical. That is the simulated view is computationally indistinguishable from the actual execution view. Besides, for specific privacy, since the semantic security of Distributed Two Trapdoors Public-Key cryptosystem has been proven in [35], the random number from a sufficiently large domain generated by C_1 and the random intermediate results transmitted in this protocol guarantee the privacy of data, query, and result. Due to the permutation function π, the adversary cannot trace back to the corresponding data records, which preserves the access patterns privacy. In addition, note that, according to the two packed values V_0 and V_1 stored in VSI, it guarantees that the server cannot know the order relationship between the query point and the points in the dataset while the random number r_0 guarantees the trapdoor unlinkability. To sum up, the SMD$_n$ protocol is secure.

Similarly, we can prove that the SDC, SGC, and SCR protocols are secure under our security model. Thus, we can obtain the following theorem.

Theorem 4: The MSV k NN is secure if DT-PKC is semantically secure.

Proof: Based on the Theorem 3, we can get that each sub-protocol involved in MSV k NN protocol is secure, and meanwhile, according to Theorem 2, it is easy to prove that the MSV k NN protocol is secure.

Theorem 5: (Correctness of MSV k NN) If the hash value of all neighbors of each resulting point and the signature of each resulting point can be rebuilt correctly, the query result is correct.

Proof: As proven in existing works, the underlying hash function is collision-resistant and the correctness can be guaranteed by the signature of each resulting point. Hence, the key point is to prove the signature of each resulting point can be rebuilt and is correct. After receiving the result set $\mathcal{R}$ and verification object $\mathcal{VO}$, the user can first obtain the correct signature Sig of each resulting point by decrypting it using his/her private key sk_u from $\mathcal{VO}$. Then, if there exist a tampered point $\mathcal{P} \in \mathcal{R}$ but $\mathcal{P} \notin \mathcal{D}$, the user can rebuilt the signature $\widehat{Sig(\mathcal{P})}$ of point $\mathcal{P}$ by computing the hash value $\mathcal{H}(\mathcal{NVC}(\mathcal{P}))$ of its neighbors from $\mathcal{VO}$ and the hash value $\mathcal{H}(\mathcal{P})$ of point $\mathcal{P}$ from $\mathcal{R}$. Further, due to the collision resistance of the hash function, $\widehat{Sig(\mathcal{P})}$ must not be identical with $Sig(\mathcal{P})$. Therefore, Theorem 5 has been proved.

Theorem 6: (Completeness of MSV k NN) If the distance between each resulting point and the query is actually less than the distances between other points and the query, the query results are complete.

Proof: Based on the properties of the Voronoi diagram, we can easily check this theorem. First, according to Theorem 5, all resulting points and their neighbors are correct. Then, following Property 1, if the distance between the nearest neighbor point $\mathcal{P}_1$ and the query is no more than its neighbors' distances, the nearest neighbor point is complete. Next, following Property 2, the distances of resulting points $\mathcal{P}_i (2 \leq i \leq k)$ is no more than the distances of points in $\mathcal{NVC}(\mathcal{P}_1) \cup \ldots \cup \mathcal{NVC}(\mathcal{P}_{i-1}) - \mathcal{P}_1 \cap \ldots \cap \mathcal{P}_i$, these resulting points are complete. To sum up, the Theorem 6 has been proved.

VI. Experimental Evaluation

In this section, we evaluate the performance of the proposed secure minimum distance protocol SMD compared with SMIN [10], SMS [15] and SMC [30]. Next, under different parameter settings, we also evaluate the performance of our proposed MSV k NN compared with SV k NN [30], SecEQP [2], Sec k NN [33], S k NN$_A$ [10] and S k NN$_B$ [11]. In addition, we also report the performance of the proposed verification scheme.

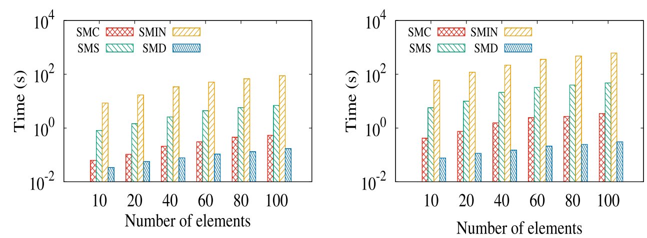
A. Experiment Setup

Datasets: In our experiment, we adopt a real dataset Foursquare and a synthetic dataset SYN. Specifically, the Foursquare dataset is a collection of Foursquare check-ins in Tokyo. It consists of 573,703 check-ins occurring in 67,123 venues with latitude and longitude. SYN dataset is generated randomly with uniform distribution, which contains 10K two-dimensional points with the length of 12 bits. For ease of computation, we transfer the geodetic coordinates to plane coordinates by Miller projection and standardize them to integers with 12 bits length.

Parameter Setting: We measure the performance of the algorithms by varying the number of points n from 2,000 to 10,000, the query parameter k from 1 to 20, the key size SK from 512 to 1024, the security parameter κ from 10 to 20 and the grid size m from 16 to 64. The detailed setting is shown in Table VI.

TABLE VI
Parameter Settings (Bold Values are Default Values)

Name	Setting
# of points n	**2,000** 4,000 6,000 8,000 10,000
query k	**1** 5 10 15 20
key size SK	512 **1024**
security parameter κ	**10** 13 15 18 20
grid granularity m	16 **32** 64



(a) SK = 512, κ = 10 (b) SK = 1024, κ = 10

Fig. 5. Computation time of secure minimum computation on Foursquare.

Setup: We implement all the above algorithms in Java and perform experiments on a PC with 8-core Intel(R) Core(TM) i7-6700 3.40GHz CPU and 40GB RAM running Windows 10. In addition, we utilize the Distributed Two Trapdoors Public-Key cryptosystem [16] as the encryption function and SHA-1 as the hash function.

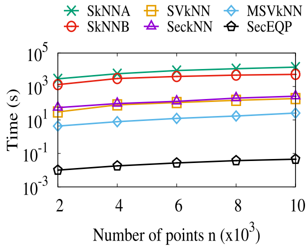
B. Evaluation of Secure Minimum Computation

In this subsection, with the varied key size SK over different datasets, we compare the performance of our proposed secure minimum distance protocol with the state-of-the-art protocols SMIN [10], SMS [15], and SMC [30], where SMS [15] cannot support access patterns privacy.

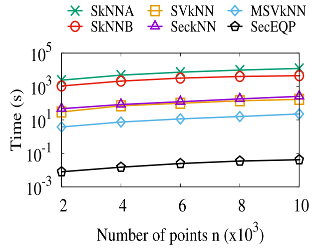
As shown in Fig. 5, the time cost of our proposed approach is significantly lower than SMIN, SMS, and SMC. Specifically, in Fig. 5(a), our proposed approach SMD at least runs about 250, 23, and 2 times faster than SMIN, SMS, and SMC when $SK = 512$. For example, when the number of elements is 10, the time of SMD takes about 0.034 s, but SMIN, SMS, and SMC consume 8.5 s, 0.81 s, and 0.063, respectively. Further, when $SK = 1024$, as shown in Fig. 5(b), our proposed approach runs about 700, 73, and 5 times faster than SMIN, SMS, and SMC, respectively. Besides, note that our SMD protocol integrates the functionality of distance computation and minimum computation, which avoids extra secure distance protocol, but other protocols need to obtain distances first.

C. Evaluation of Secure kNN Computation

In this part, we study the performance of secure kNN computation by comparing our approach MSV k NN with the state-of-the-art approaches SecEQP [2], S k NN$_A$ [10], S k NN$_B$ [11], Sec k NN [33] and SV k NN [30] in different parameter settings, where SecEQP, S k NN$_A$, S k NN$_B$ and SV k NN

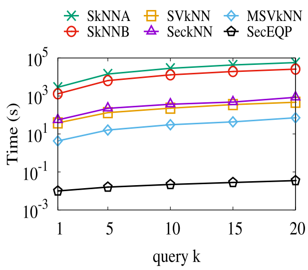


(a) Varying n on Foursquare

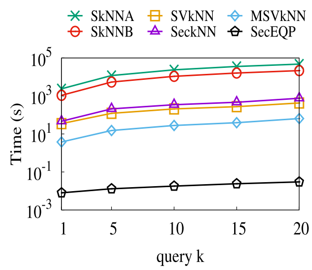


(b) Varying n on SYN

Fig. 6. The impact of varying dataset size n.

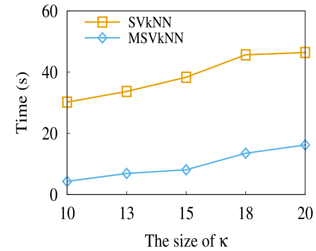


(a) Varying k on Foursquare

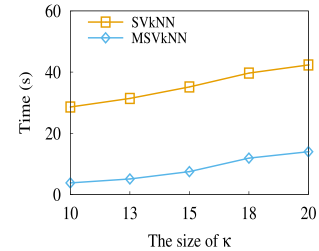


(b) Varying k on SYN

Fig. 7. The impact of varying query k.

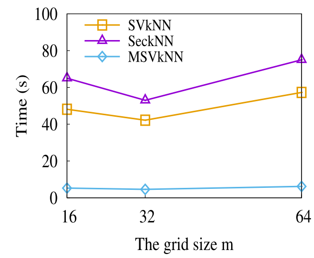


(a) Varying κ on Foursquare

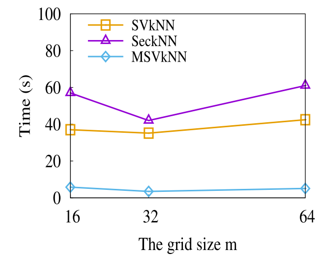


(b) Varying κ on SYN

Fig. 8. The impact of varying security parameter κ.



(a) Varying m on Foursquare



(b) Varying m on SYN

Fig. 9. The impact of varying grid size m.

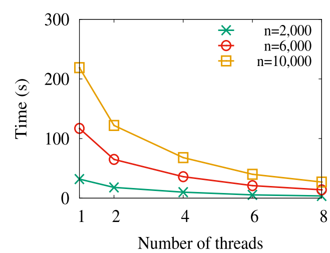


Fig. 10. Number of threads versus Query delays.

cannot support multi-user setting, and SecEQP, S k NN$_A$, S k NN$_B$ and Sec k NN cannot support result verification.

Impact of Varying n: In Fig. 6, overall, the five algorithms consume the time in a linear trend with the increase of dataset size n. But our proposed method is more efficient than other algorithms. In detail, in Fig. 6(a), comparing with S k NN$_A$, S k NN$_B$, Sec k NN and SV k NN, when n = 10,000, our method MSV k NN (25.9 s) only takes about 0.18 %, 0.49 %, 9.8 % and 13.6 % time cost of S k NN$_A$ (14.5$\times 10^3$ s), S k NN$_B$ (5.4$\times 10^3$ s), Sec k NN (0.26$\times 10^3$ s) and SV k NN (0.19$\times 10^3$ s), respectively. This is because our SMD protocol is more efficient and takes less encryption and decryption operations. Obviously, SecEQP has the best performance since it resorts to cryptographic hash but cannot support multi-user setting and result verification. Also, a similar performance trend can be seen on SYN, as shown in Fig. 6(b).

Impact of Varying k: Fig. 7 shows the performance of the algorithms with the varied k. Specifically, in Fig. 7(a), the results indicate that the time cost of these algorithms increases linearly with k increasing. Note that, our approach is also obviously better than others. Especially, when k = 20, our MSV k NN (71.3 s) only takes about 0.12 %, 0.27 %, 8.3 % and 15.2 % time cost of S k NN$_A$ (57.6$\times 10^3$ s), S k NN$_B$ (26.1$\times 10^3$s), Sec k NN (854 s) and SV k NN (468 s), respectively. Also, SecEQP has the best performance for the same reason. Meanwhile, as can be seen from Fig. 7(b), the performance on SYN is similar to the performance on Foursquare.

Impact of Varying κ: In this experiment, due to κ not involved in S k NN$_A$, S k NN$_B$ and Sec k NN, as shown in Fig. 8, we evaluate the performance of our proposed approach and SV k NN with varying κ. We can see that the growth of the search time is linear inconspicuously. In addition, we can see that our scheme have better performance. This is because our scheme adopts a more compact and efficient index and our secure protocols are also more efficient. For example, in the best case, our MSV k NN is 7 times faster than SV k NN in Fig. 8(a).

Impact of Varying m: Fig. 9(a) and (b) show the time cost of our proposed algorithm with Sec k NN and SV k NN on varying m on Foursquare and SYN, respectively. Here, Sec k NN is not involved in m and we apply grid partition to Sec k NN. We can see that, when $m = 32$, the time cost of all algorithms achieves the minimum. This is because the number of cells in a grid decreases with m increasing, and when $m = 32$, this situation reaches equilibrium. For example, when $m = 32$, the time cost of Sec k NN, SV k NN, and MSV k NN is 53 s, 42 s, and 4.6 s, respectively.

Impact of Parallelism: To further improve the query cost and show the performance of the larger key size (i.e., 2,048 bits), we implement our scheme by adopting parallelism with varying dataset sizes and key size $SK = 2,048$. More specifically, we divide the partitioned grids and buckets into sub-sets and each sub-set is assigned to a thread to calculate the SGC and SCR protocols. Here, this parallel scheme is implemented through multi-threading and the result is shown in Fig. 10. We can see that

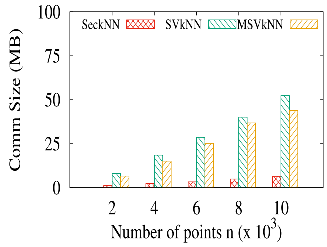
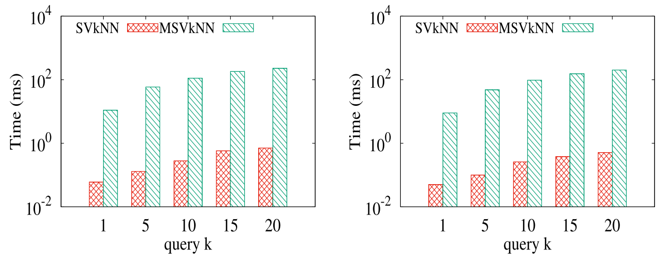
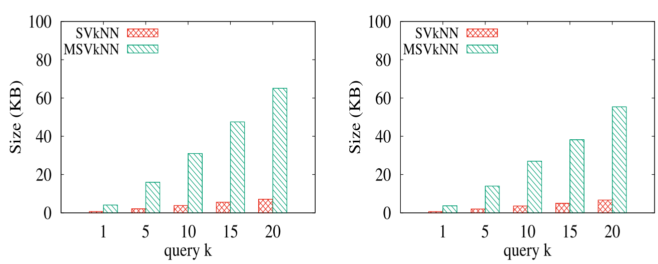


Fig. 11. Dataset size n versus Communication cost.



(a) Varying k on Foursquare (b) Varying k on SYN

Fig. 12. Varying query k versus Verification time.



(a) Varying k on Foursquare (b) Varying k on SYN

Fig. 13. Varying query k versus $\mathcal{VO}$ size.

with the number of threads increasing, the time cost decreases apparently.

D. Evaluation of Communication

In this subsection, we evaluate the communication overhead between cloud C_1 and C_2 during the entire query processing. As shown in Fig. 11, it presents the communication overhead of MSV k NN compared with Sec k NN and SV k NN with varying dataset sizes. We can observe that Sec k NN has the lowest cost because it fails to preserve the access pattern while cannot support result verification. And the experimental result is in agreement with the theoretical analysis.

E. Evaluation of Result Verification

Last, along with varying k, we evaluate the performance of the result verification of MSV k NN and SV k NN over Foursquare and SYN. Specifically, Fig. 12 shows the processing time on the user side w.r.t. the query k and Fig. 13 shows the verification object size from the cloud server w.r.t. the query k.

As shown in Fig. 12, the verification time is lower for SYN accordingly since the number of neighbors for one point of SYN is less than that of Foursquare. Moreover, the verification time of SV k NN is less than MSV k NN. This is because the information in the result set and verification object of MSV k NN is in the form of ciphertext, which needs to decrypt by the user using his/her own private key. But the time cost of our method is still acceptable. For instance, when k varies from 1 to 20, the verification time of SV k NN and MSV k NN grows from 0.06 ms and 11 ms to 0.71 ms and 229 ms, respectively. Regarding the verification object $\mathcal{VO}$ size, Fig. 13 shows that the $\mathcal{VO}$ size of SYN is also less than that of Foursquare and the verification object size of SV k NN is lower than MSV k NN for the same reason. That is the information of the verification object in MSV k NN is encrypted. For instance, when k varies from 1 to 20, the verification object size of SV k NN and MSV k NN grows from 0.69 KB and 4.1 KB to 7.11 KB and 65.1 KB, respectively.

VII. Related Work

In this section, we introduce the related work from the following three aspects: i.) secure kNN query, ii.) verifiable query and iii.) multi-user query.

Secure kNN Query: Secure kNN query as a hot topic has been widely studied in recent years. Wong et al. proposed a novel encryption scheme Asymmetric-Scalar-Product-Preserving Encryption (ASPE) to compute the distance in a private manner [3], [4]. However, this encryption scheme has been demonstrated to be insecure under the known-plaintext attack [19]. Afterward, Lei et al. [14] resorts to SSE scheme with Bloom filter to design secure index for efficient kNN search. Meanwhile, Choi et al. [5] and Wang et al. [13] adopted mutable order-preserving encryption (OPE) [6] that improves security. Unfortunately, all of these methods disclose the access patterns privacy. To avoid access patterns leakage, Elmehdwi et al. [10] and Kim et al. [11] adopted two servers to perform the secure protocols collaboratively using paillier homomorphic encryption. However, the performance cannot be acceptable due to the high computational cost. In addition, Yao et al. [19], Wang et al. [20] and Li et al. [12] have proposed approaches to deal with secure NN query, but both kNN scenario and the access patterns privacy are not supported. Yi et al. [21] and Lei et al. [2] have addressed secure approximate kNN query, but their work cannot be applied to the exact query. And Yu et al. [40] have studied secure top-k query considering both spatial and textual conditions. However, all of them still suffer from access patterns privacy.

Verifiable Query: The framework of Merkle Hash Tree (MHT)[22] is widely used to verify location-based queries integrating spatial indexes. For instance, MB-tree and MR-tree are designed for verifying one-dimension and multi-dimensional spatial queries respectively (e.g., range query [23], kNN query [24], top-k query [25], and skyline query [26]). However, following this tree-based authentication structure, the access patterns privacy will be inevitably exposed to the attacker. In addition, aggregation and chaining [27] is another way to verify the integrity of query results. This scheme links the signatures of adjacent data values to ensure no result can be omitted. However, following this way, the user has to know the boundary data, which leads to the failure of data privacy protection. Moreover, for the secure and verifiable query, Rong et al. [18] and Cui et al. [30] study the secure and verifiable kNN query. But, the former has lower security and the latter has lower efficiency. Besides, Wu et al. [1] have first solved the problem of secure and verifiable range query. Unfortunately, their approach cannot be applied to our secure and verifiable kNN problem and also suffers from revealing access patterns privacy.

Multi-User Query: The techniques for multi-user query have been proposed by researchers recently and are still being studied. Liu et al. [35] proposed a Distributed Two Trapdoors Public-Key Cryptosystem to support different users with different keys during the query process. Based on this, Cheng et al. [34] and Nayak et al. [36] have explored range query and keyword query, respectively. But, both of them cannot support result verification and suffer from high search overhead. In addition, to improve the performance, Song et al. [31] and Liu et al. [37] have explored multi-user multi-keyword search using proxy re-encryption and symmetric encryption, respectively. Han et al. [32] have studied multi-party record linkage. However, all of them cannot be applied to the kNN query directly and also cannot support result verification. Note that, only one work [33] has been proposed to solve the issue of multi-user and secure kNN query. Unfortunately, this work cannot support result verification as well and suffers from higher query overhead.

In summary, all of the above works cannot satisfy the requirements of privacy, result verification, and multi-user setting completely and cannot be applied to address our problem. Therefore, it is of great significance to deal with the problem of multi-user, verifiable, secure kNN query.

VIII. Conclusion

In this article, we investigated and proposed multi-user, secure and verifiable k nearest neighbours query (MSV k NN). The goal of MSV k NN is to search accurate results while preserving the privacy of the data, the query, the result and access patterns and guaranteeing the correctness and completeness of the results under multi-user setting. To this end, we first proposed a verifiable and secure index (VSI) to support private search and result verification for multiple users. After that, we designed a series of novel secure protocols and a compact verification strategy to facilitate the operation over VSI. At last, comprehensive analysis and experiments show that our schemes are efficient and practical under different settings. In our future work, we will focus on the another important characteristic of database query, i.e., dynamic update. In addition, we will pay more effort on the trade-off between security and efficiency.

References

[1] S. Wu, Q. Li, G. Li, D. Yuan, X. Yuan, and C. Wang, "ServeDB: Secure, verifiable, and efficient range queries on outsourced database," in *Proc. IEEE Int. Conf. Data Eng.*, 2019, pp. 626–637.

[2] X. Lei, A. X. Liu, R. Li, and G. Tu, "SecEQP: A secure and efficient scheme for kNN query problem over encrypted geodata on cloud," in *Proc. IEEE Int. Conf. Data Eng.*, 2019, pp. 662–673.

[3] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM Int. Conf. Manage. Data*, 2009, pp. 139–152.

[4] Y. Zhu, R. Xu, and T. Takagi, "Secure k-NN computation on encrypted cloud data without sharing key with query users," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 55–60.

[5] S. Choi, G. Ghinita, H. S. Lim, and E. Bertino, "Secure kNN query processing in untrusted cloud environments," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 11, pp. 2818–2831, Nov. 2014.

[6] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE Int. Conf. Secur. Privacy*, 2013, pp. 463–477.

[7] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan, "Private queries in location based services: Anonymizers are not necessary," in *Proc. ACM Int. Conf. Manage. Data*, 2008, pp. 121–132.

[8] P. Williams, R. Sion, and B. Carbunar, "Building castles out of mud: Practical access pattern privacy and correctness on untrusted storage," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2008, pp. 139–148.

[9] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. Int. Conf. Netw. Distrib. Syst. Secur. Symp.*, 2012, pp. 1–15.

[10] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE Int. Conf. Data Eng.*, 2014, pp. 664–675.

[11] H. I. Kim, H. J. Kim, and J. W. Chang, "A secure kNN query processing algorithm using homomorphic encryption on outsourced database," *Data Knowl. Eng.*, vol. 123, 2019, Art. no. 101602.

[12] R. Li, A. X. Liu, H. Xu, Y. Liu, and H. Yuan, "Adaptive secure nearest neighbor query processing over encrypted data," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 91–106, Jan./Feb. 2022.

[13] B. Wang, Y. Hou, and M. Li, "QuickN: Practical and secure nearest neighbor search on encrypted large-scale data," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 2066–2078, Third Quarter 2022.

[14] X. Lei, G. -H. Tu, A. X. Liu, and T. Xie, "Fast and secure kNN query processing in cloud computing," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2020, pp. 1–9.

[15] A. Liu, K. Zheng, L. Li, G. Liu, L. Zhao, and X. Zhou, "Efficient secure similarity computation on encrypted trajectory data," in *Proc. IEEE Int. Conf. Data Eng.*, 2015, pp. 66–77.

[16] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.

[17] A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *Proc. IEEE Int. Conf. Found. Comput. Sci.*, 1986, pp. 162–167.

[18] H. Rong, H. Wang, J. Liu, W. Wu, and M. Xian, "Efficient integrity verification of secure outsourced kNN computation in cloud environments," in *Proc. IEEE Int. Conf. Trustcom*, 2016, pp. 236–243.

[19] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Proc. IEEE Int. Conf. Data Eng.*, 2013, pp. 733–744.

[20] B. Wang, Y. Hou, and M. Li, "Practical and secure nearest neighbor search on encrypted large-scale data," in *Proc. IEEE Conf. Comput. Commun.*, 2016, pp. 1–9.

[21] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical approximate k nearest neighbor queries with location and query privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1546–1559, Jun. 2016.

[22] R. C. Merkle, "A certified digital signature," in *Proc. Int. Conf. Theory Appl. Cryptology*, 1989, pp. 218–238.

[23] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Authenticated indexing for outsourced spatial databases," in *Proc. Int. Conf. Very Large Data Bases*, 2009, pp. 631–648.

[24] M. L. Yiu, E. Lo, and D. Yung, "Authentication of moving kNN queries," in *Proc. IEEE Int. Conf. Data Eng.*, 2011, pp. 565–576.

[25] Q. Chen, H. Hu, and J. Xu, "Authenticating top-k queries in location-based services with confidentiality," in *Proc. Int. Conf. Very Large Data Bases*, 2013, pp. 49–60.

[26] X. Lin, J. Xu, and H. Hu, "Authentication of location-based skyline queries," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2011, pp. 1583–1588.

[27] H. Pang, A. Jain, K. Ramamritham, and K. Tan, "Verifying completeness of relational query results in data publishing," in *Proc. ACM Int. Conf. Manage. Data*, 2005, pp. 407–418.

[28] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *Proc. IEEE Int. Conf. Data Eng.*, 2017, pp. 633–644.

[29] C. Xu, J. Xu, H. Hu, and M. H. Au, "When query authentication meets fine-grained access control: A zero-knowledge approach," in *Proc. ACM Int. Conf. Manage. Data*, 2018, pp. 147–162.
[30] N. Cui, X. Yang, B. Wang, J. Li, and G. Wang, "SVkNN: Efficient secure and verifiable k-nearest neighbor query on the cloud platform," in *Proc. IEEE Int. Conf. Data Eng.*, 2020, pp. 253–264.
[31] F. Song, Z. Qin, J. Liang, and X. Lin, "An efficient and privacy-preserving multi-user multi-keyword search scheme without key sharing," in *Proc. IEEE Int. Cryptol. Conf.*, 2021, pp. 1–6.
[32] S. Han, D. Shen, T. Nie, Y. Kou, and G. Yu, "Private blocking technique for multi-party privacy-preserving record linkage," *Data Sci. Eng.*, vol. 2, pp. 187–196, 2017.
[33] K. Cheng et al., "Secure k-NN query on encrypted cloud data with multiple keys," *IEEE Trans. Big Data*, vol. 7, no. 4, pp. 689–702, Oct. 2021.
[34] K. Cheng, Y. Shen, Y. Wang, L. Wang, and J. Ma, "Strongly secure and efficient range queries in cloud databases under multiple keys," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2494–2502.
[35] X. Liu, H. D. Robert, and R. C. Kim-Kwang, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 11, pp. 2401–2414, Nov. 2016.
[36] S. K. Nayak and S. Tripathy, "SEMKC: Secure and efficient computation over outsourced data encrypted under multiple keys," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 1, pp. 414–428, First Quarter 2021.
[37] X. Liu, G. Yang, Y. Mu, and R. H. Deng, "Multi-user verifiable searchable symmetric encryption for cloud storage," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 6, pp. 1322–1332, Nov./Dec. 2020.
[38] Y. Yang, X. Liu, and R. H. Deng, "Multi-user multi-keyword rank search over encrypted data in arbitrary language," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 2, pp. 320–334, Mar./Apr. 2020.
[39] X. Ding, Z. Wang, P. Zhou, K. -K. R. Choo, and H. Jin, "Efficient and privacy-preserving multi-party skyline queries over encrypted data," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4589–4604, Aug. 2021.
[40] X. Yu, Y. Hu, R. Zhang, Z. Yan, and Y. Zhang, "Secure outsourced top-k selection queries against untrusted cloud service providers, in *Proc. IEEE/ACM 29th Int. Symp. Qual. Serv.*, 2021, pp. 1–10.

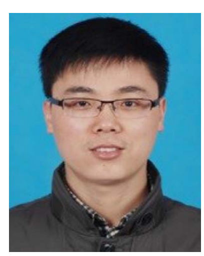
Ningning Cui received the master's degree in computer science from the Inner Mongolia University of Science and Technology, China, in 2015, and the PhD degree from Northeastern University, China, in 2020. He is currently a lecture with Anhui University. His research interests include data privacy, query processing, information security.

Kang Qian received the bachelor's degree in computer science from the Anhui University of Technology, China, in 2020. He is currently working towards the master's degree with Anhui University. His research interests include data privacy, query processing, information security.

Taotao Cai received the PhD degree in computer science from Deakin University, Australia, in 2020. He is currently a postdoctoral research fellow with the School of Computing, Macquarie University. Before moving to Macquarie, he worked as an associate research fellow with the School of Info Technology, Deakin University. His research interests include graph data processing, social network analytic computing, and data mining.

Jianxin Li (Member, IEEE) received the PhD degree in computer science from the Swinburne University of Technology, Australia, in 2009. He is an associate professor in data science with the School of Information Technology, Deakin University, Australia. His research interests include graph database query processing and optimization, social network analytics and computing, complex network representation learning, and personalized online learning analytics. He is also a grant assessor in Australia Research Council Discovery Programs and Linkage Programs, and serves as invited reviewers for many top journals and program committee members in many top conferences.

Xiaochun Yang (Member, IEEE) received the PhD degree from Northeastern University, China, in 2001, in computer software and theory. She is currently a professor with the School of Computer Science and Engineering, Northeastern University, China. Her research interests include Big Data management and knowledge engineering, data quality management, data privacy preserving, and recommender system. More details about her research can be found at http://faculty.neu.edu.cn/yangxiaochun.



Jie Cui (Senior Member, IEEE) received the PhD degree from the University of Science and Technology of China, in 2012. He is currently a professor and PhD supervisor of the School of Computer Science and Technology, Anhui University. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). He has more than 150 scientific publications in reputable journals, academic books and international conferences.

Hong Zhong received the PhD degree in computer science from the University of Science and Technology of China, in 2005. She is currently a professor and PhD supervisor of the School of Computer Science and Technology, Anhui University. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). She has more than 200 scientific publications in reputable journals, academic books and international conferences.