Computer Standards & Interfaces 97 (2026) 104112

Contents lists available at ScienceDirect

Computer Standards & Interfaces

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

Efficient and secure multi-user *k*NN queries with dynamic POIs updating

Yining Jia [a,b,c], Yali Liu [a,b,c,*], Congai Zeng [a,b,c], Xujie Ding [a,b,c], Jianting Ning [d,e]

[a] School of Artificial Intelligence and Computer Science, Jiangsu Normal University, Xuzhou, Jiangsu Province, 221116, China
[b] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu Province, 210023, China
[c] Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi Province, 541004, China
[d] School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei Province, 430072, China
[e] Faculty of Data Science, City University of Macau, 999078, Macao Special Administrative Region of China

ARTICLE INFO

Keywords:
Cloud computing
Security
*k*NN queries
Dynamic POIs updating

ABSTRACT

The *k*-nearest neighbors (*k*NN) query is a key operation in spatial and multimedia databases, which is widely applied in fields such as electronic healthcare and Location-Based Services (LBS). With the rapid development of cloud computing, uploading private data of Data Owner (DO) to Cloud Servers (CS) has become a trend. However, existing *k*NN queries schemes are not designed for multi-user environments, cannot timely update the points of interest (POIs) stored in CS, and suffer from low query efficiency. Therefore, this paper proposes efficient and secure multi-user *k*NN queries with dynamic POIs updating, named DESM*k*NN, which achieves secure multi-user *k*NN queries. To improve query efficiency, DESM*k*NN adopts a two-stage search framework, which consists of an initial filtering stage based on hierarchical clustering to effectively constrain the search range, followed by a more efficient precise search stage. Based on this framework, DESM*k*NN designs a set of security protocols for efficient query processing and enables dynamic POIs updates. Meanwhile, DESM*k*NN not only utilizes Distributed Two Trapdoors Public-Key Cryptosystem (DT-PKC) to enable multi-user queries but also ensures data privacy, query privacy, result privacy and access pattern privacy. Moreover, DESM*k*NN can verify the correctness and completeness of queries results. Finally, security analysis proves that DESM*k*NN meets the formal security definition of multiparty computation, and experimental evaluation shows that DESM*k*NN improves query efficiency by up to 45.5% compared with existing *k*NN queries scheme.

1. Introduction

LBS [1–3] are increasingly integrated into real-world applications, such as ride-hailing platforms (e.g., Uber, DiDi), navigation systems (e.g., Google Maps, Baidu Maps), and online food delivery services. These services heavily rely on POIs databases to provide personalized and efficient responses to queries of query user (QU). Among various query types, the *k*NN query [4,5] is one of the most fundamental methods, which aims to find the *k* nearest POIs to a given query point. With the rapid development of cloud computing [6,7], DO increasingly outsource their POIs databases to CS, which provides scalable storage and massive computing resources. Well-known commercial platforms, such as Amazon Web Services and Google Cloud Platform, already provide such services to support efficient *k*NN queries in LBS. Although outsourcing databases to CS improves data accessibility and flexibility, it makes data more susceptible to unauthorized access threats. In practice, POIs often contain sensitive or private information. For instance, POIs databases may include the locations of hospitals, government facilities, or user-related activity areas in intelligent transportation and LBS systems. Once such information is exposed, it can lead to privacy leakage, commercial losses, or even public security risks [4]. Therefore, to protect POIs from malicious access or theft by CS and unauthorized users, DO needs to encrypt them before outsourcing to CS. In addition, security needs to be considered in query processing to maintain efficiency and protect the confidentiality of POIs databases.

Although *k*NN queries have been widely studied in recent years, several limitations still hinder their applicability in practice. First, most existing schemes [8,9] for *k*NN queries are based on static spatial data [10], where the database remains unchanged within a certain time interval. Consistent with this common setting, DESM*k*NN also assumes that POIs are static during query processing to enable fair performance comparison. However, in practice, POIs may change over time, and their insertion or deletion frequency varies across different areas because these updates are driven by real-world change. In rapidly developing areas where new facilities emerge or existing ones close frequently, POI updates occur more frequently, whereas in more stable regions, such updates tend to be infrequent. This dynamic updates of

* Corresponding author at: School of Artificial Intelligence and Computer Science, Jiangsu Normal University, Xuzhou, Jiangsu Province, 221116, China.
E-mail address: liuyali@jsnu.edu.cn (Y. Liu).

https://doi.org/10.1016/j.csi.2025.104112
Received 12 June 2025; Received in revised form 18 November 2025; Accepted 8 December 2025
Available online 11 December 2025

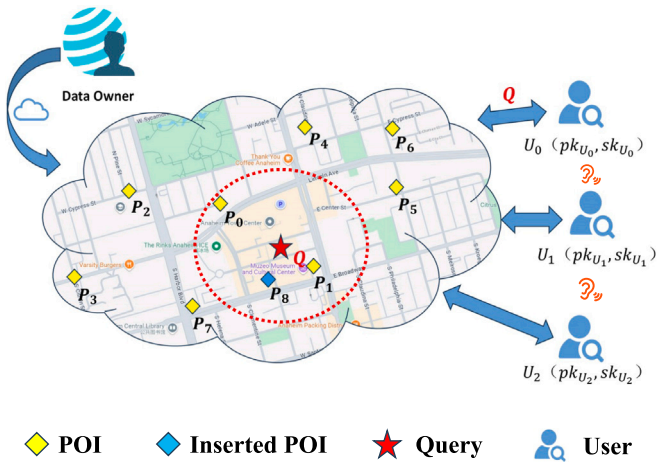


Fig. 1. Sample of the kNN query ($k = 2$).

POIs reflects the continuous changes in the physical environment. As shown in Fig. 1, U_0 searches for the two nearest neighbors ($k = 2$) in a POIs database $D = \{p_0, \ldots, p_7\}$. The original 2NN query Q was $\{p_0, p_1\}$. When a new and closer point p_8 is inserted, the correct 2NN result becomes $\{p_1, p_8\}$. This example shows that any updates to the POI database, such as the insertion, modification, or deletion of POIs, may change the query results. Therefore, dynamic updates must be supported in outsourced POI databases. Second, existing schemes mostly use Asymmetric-Scalar-Product-Preserving Encryption (ASPE) [11,12] or pure homomorphic encryption algorithms to encrypt outsourced data. Unfortunately, ASPE has been demonstrated to be insecure under the known-plaintext attacks [13], and homomorphic operations lead to a significant computational cost. These limitations raise the challenge of designing an efficient and secure query mechanism. Finally, most solutions [14,15] assume a single-user setting, where all QUs share the same secret key to enable computability of encrypted data across multi-user. In practice, the assumption of single-user setting has obvious flaws. Once the unique key of any QUs is leaked, the entire encrypted database can be completely decrypted, and the query content may also be intercepted by the adversary. As illustrated in Fig. 1, in such a single-user setting, U_1 and U_2 can capture the query content and result of U_0 and decrypt them using the same secret key as U_0. This highlights the need for secure multi-user queries.

To resolve the aforementioned challenges, this paper proposes DESMkNN. The contributions of DESMkNN are as follows:

(1) ***Dynamic POIs Updating***: DESMkNN innovatively designs secure insertion and deletion protocols, which avoids the problem of incorrect and incomplete query results.
(2) ***Efficient Query***: DESMkNN proposes an efficient two-stage search framework, which improves the query performance.
(3) ***Multi-User Query***: DESMkNN designs a series of secure protocols based on DT-PKC, which achieves secure multi-user kNN queries.
(4) ***Security*** & ***Performance***: Security analysis shows that the proposed DESMkNN is secure. Additionally, experimental evaluation shows that DESMkNN improves query efficiency by up to 45.5% compared with existing kNN queries scheme on two real datasets (California Road Network and Points of Interest, San Francisco Road Network[1]).

The rest of this paper is structured as follows. Section 2 presents related work. Section 3 describes preliminaries. The architecture and security model of DESMkNN is defined in Section 4. In Section 5, the system construction is introduced. Section 6 presents the specific query procedure for DESMkNN. Next, Section 7 analyzes computational complexity, communication complexity, and security. Section 8 provides an experimental evaluation of DESMkNN. Section 9 concludes this paper.

2. Related work

Secure Key-Sharing Query: Wong et al. [11] introduced a kNN queries scheme for encrypted data based on ASPE. However, ASPE relied on a secret matrix to transform data points and query points, which required secret key to be shared among all QUs and DO. Additionally, ASPE has been proven insecure against known-plaintext attacks [13]. To enhance query security, Elmehdwi et al. [15] developed a set of two-party computation protocols based on the Paillier cryptosystem. Although scheme [15] preserved the privacy of query results, QUs hold DO's private key, and the query efficiency remains low. Moreover, scheme [16] employed Delaunay triangulation and order-preserving encryption [18] to accurately solve the secure kNN problem. Nevertheless, the encryption schemes in [16] are symmetric, which also required DO and QUs to share the key. Cui et al. [8] proposed an efficient, secure, and verifiable kNN queries scheme, which employed a secure index structure to ensure data security and result integrity, along with a set of novel protocols and verification strategies for various index operations. However, the search complexity of scheme [8] was linearly related to the database size, which led to a lack of scalability. To address the efficiency issues in [8], Liu et al. [14] introduced a two-stage search framework for secure and verifiable kNN queries, which integrated Edge Servers (ES) into the classic Twin-Cloud model by leveraging adaptive encryption strategies and secure data partitioning to optimize query performance. However, both scheme [8] and scheme [14] could not resolve the key-sharing issue.

Secure Multi-User Query: To support multi-user kNN queries, researchers first focused on multi-key queries. Cheng et al. [17] implemented kNN queries with multi-key support, where DO and QUs had their own keys, and each QU's key was not shared with others. However, scheme [17] incurred high computational cost and lacked result verification. Subsequently, Liu et al. proposed the DT-PKC [19], which also allowed different QUs to use different keys during queries. Building on the DT-PKC, Cheng et al. [20] and Nayak et al. [21] explored range queries and keyword queries, respectively. Nevertheless, scheme [20] and scheme [21] still suffered from computational cost and the inability to verify results. Cui et al. [9] introduced a method for secure and verifiable kNN queries by utilizing DT-PKC, which encrypted grid and bucket divisions within the Voronoi diagram to maintain data security, while also introducing a verification strategy to ensure the correctness and completeness of the query results. However, scheme [9] relied heavily on homomorphic encryption and data packing techniques, which led to high computational cost and search complexity. Moreover, scheme [9] fails to address the issue of dynamic updates for POIs.

In summary, the limitations in the existing kNN queries schemes are as follows: (1) The single-user queries schemes have a risk of key leakage. (2) The multi-user queries schemes have low efficiency. (3) Most existing queries schemes unable to achieve dynamic updates of POIs. For ease of exhibition, we summarize the above works in Table 1.

3. Preliminaries

3.1. Voronoi diagram

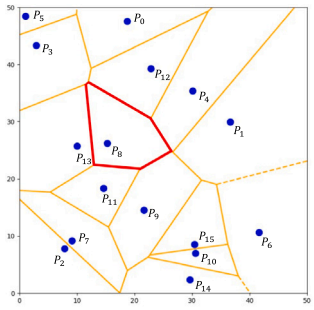
The Voronoi diagram [22] partitions the plane according to a set of points. Each Voronoi Cell (VC) corresponds to a point and contains all locations that are closer to this point than to any other. Two points are Voronoi neighbors if their cells share an edge, and the neighbor set of a point is denoted as $VN(p)$.

[1] https://users.cs.utah.edu/~lifeifei/SpatialDataset.htm.

Table 1
Summary of existing kNN query works.

Method	Data privacy	Query privacy	Result privacy	Access patterns	Verifiable	Multi-user	POIs updating
Wong [11]	×	√	√	×	×	×	×
Elmehdwi [15]	√	√	√	√	×	×	×
Choi [16]	√	√	√	×	×	×	×
Cheng [17]	√	√	√	×	×	√	×
Cui [8]	√	√	√	√	√	×	×
Liu [14]	√	√	√	×	√	×	×
Cui [9]	√	√	√	√	√	√	×

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



(a) Voronoi diagram

POI	Coordinate	Voronoi Neighbors
P_0	(19,48)	$\{P_3, P_4, P_5, P_{12}\}$
P_1	(37,30)	$\{P_4, P_6, P_9, P_{15}\}$
P_2	(8,8)	$\{P_7\}$
...	...	...
P_{13}	(10,26)	$\{P_3, P_7, P_8, P_{11}\}$
P_{14}	(30,2)	$\{P_6, P_7, P_9, P_{10}\}$
P_{15}	(31,9)	$\{P_1, P_{10}, P_6, P_9\}$

(b) Dataset

Fig. 2. An example of Voronoi diagram.

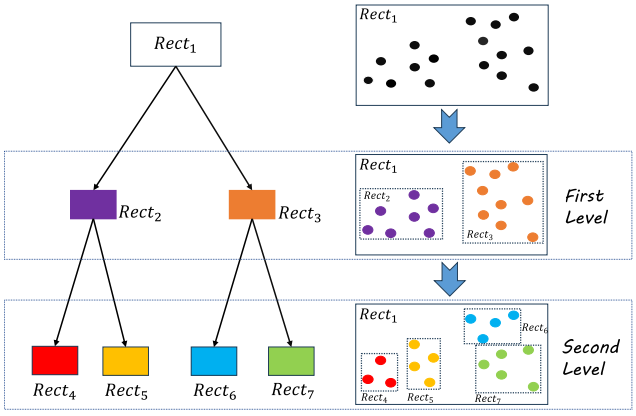


Fig. 3. R-tree structure based on hierarchical clustering.

For example, given a dataset D that contains 16 POIs as shown in Fig. 2-(b), the Voronoi diagram is shown in Fig. 2-(a). Since $VC(p_8)$ shares a common edge with $VC(p_i)$ for $i \in \{3, 4, 9, 11, 12, 13\}$, the Voronoi neighbors of p_8 include $VN(p_8) = \{p_3, p_4, p_9, p_{11}, p_{12}, p_{13}\}$. Therefore, the search result of a 3NN query is $Result = \{p_9, p_{11}, p_{13}\}$.

The Voronoi diagram has two useful properties for kNN verification:

(1) Given a query point q, the nearest neighbor of q is data point p, if $q \in VC(p)$.
(2) If data points $p_1, \ldots, p_k$ are the $k(k > 1)$ nearest neighbors of the query point q, then p_i belongs to $VN(p_1) \cup \cdots \cup VN(p_{i-1})$, for $i = 2, \ldots, k$.

3.2. R-tree index based on hierarchical clustering

The R-tree index [23] organizes spatial objects into nested rectangles, known as Minimum Bounding Rectangles, to enable efficient querying of spatial data, such as range queries [24] and nearest neighbor searches. However, the efficiency of the R-tree strongly depends on how the data are grouped during construction. To address this, DESMkNN introduces hierarchical clustering, which improves both the organization of spatial objects and the performance of query processing.

As shown in Fig. 3, it presents an R-tree with a fanout of $f = 2$, which is built from the POIs in $Rect_1$. In this construction, the data are first grouped by applying hierarchical clustering based on the Euclidean distance. This process is performed in two rounds, and the resulting clusters naturally determine the partitioning of the dataset, which is then used to build the tree structure.

3.3. Distributed two trapdoors public-key cryptosystem

The DT-PKC [19] is a variant of the traditional double trapdoor decryption cryptosystem. Given a public key pk, a private key sk, and a strong private key SK, the cryptosystem supports several algorithms that enable encryption, decryption, and collaborative key operations.

First, encryption is carried out by the algorithm Enc. Given a message $p \in \mathbb{Z}_N$ and the public key pk, the algorithm outputs the ciphertext $E_{pk}(p)$. The system then allows two types of decryption:

(1) With the private key (sk), the algorithm $WDec$ takes $E_{pk}(p)$ as input and recovers p.
(2) With the strong private key (SK), the algorithm $SDec$ also decrypts $E_{pk}(p)$ to obtain p.

A distinctive feature of DT-PKC lies in the management of the strong private key. The algorithm $SkeyS$ enables the strong private key SK to be split into two partial strong private keys, SK_1 and SK_2. This splitting supports a collaborative decryption mechanism in two steps:

(1) In step 1, $PSDec1$ takes $E_{pk}(p)$ and SK_1 as input, which results in a partially decrypted ciphertext CT_1.
(2) In step 2, $PSDec2$ completes the process by using CT_1 and SK_2, which ultimately recovers p.

3.4. Advanced comparable inner product encoding

The CIPE$_s$ scheme [25] allows edges to determine whether a value lies within a query range based on encrypted data. Compared to the original CIPE scheme, CIPE$_s$ enhances security by extending query vectors into random query matrices, which makes it more resilient to chosen plaintext attacks.

CIPE$_s$ supports several key algorithms for encryption and range query evaluation. First, the key generation algorithm $GenKey$ takes a security parameter $\kappa \in \mathbb{N}$ as input and outputs a secret key sk_c. The data encryption algorithm $EncI$ encrypts a plaintext x into ciphertext $E_c(x)$ with sk_c. To perform queries, the query encryption algorithm $EncQ$ transforms a query range $Q = [b_l, b_u]$ into an encrypted range $E_c(Q)$. Finally, the calculation algorithm Cal compares the encrypted value $E_c(x)$ with the encrypted query range $E_c(Q)$ and outputs a comparison result: -1 if $x < b_l$, 1 if $x > b_u$, and 0 if $x \in [b_l, b_u]$.

4. System architecture and security model

This section introduces the system architecture and security model of DESMkNN. A summary of notations is given in Table 2.

Table 2
Summary of notations.

D	A spatial dataset that includes n points $\{P_1, \ldots, P_n\}$
VD	Voronoi diagram built from D
sk_c	The secret key for CIPE$_s$ scheme
sk_0, pk_0	The secret/public key for DO
sk_u, pk_u	The secret/public key for users
SK, SK_1, SK_2	Strong private key and partial ones
$PSDec1(SK_1, *)$	The first step of partial decryption
$PSDec2(SK_2, *, *)$	The second step of partial decryption
$Q, E_c(Q)$	A query coverage and its encrypted range
$q, E_{pk_0}(q)$	A query point and its encrypted coordinates
$P_i, E_{pk_0}(P_i)$	A POI and its encrypted coordinates
$\widehat{Tree}_R, Tree_R$	The encrypted/clear R-tree index built from D
$\widehat{PD}, PD$	The encrypted/clear preprocessed data built from VD
$\widehat{Rect}_Q, Rect_Q$	The encrypted/clear range query generated for Q
IR	The immediate result
$\widehat{Result}, Result$	The encrypted/clear result in the exact search phase
$H(*)$	A hash function
VO	The verification object

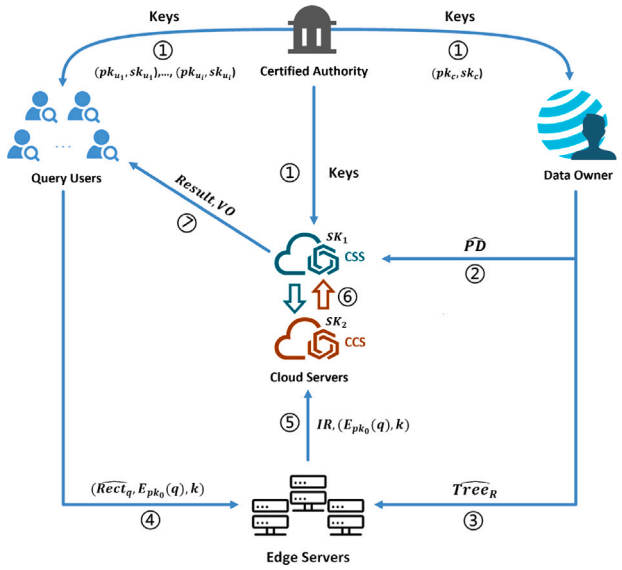


Fig. 4. System architecture.

4.1. System architecture

DESMkNN employs a two-stage framework: an initial filtering stage on ESs and a precise search stage on dual cloud servers. To protect privacy, the system adopts a dual-cloud architecture [8,9,14,26], where collusion-resilient protocols ensure both efficiency and security beyond traditional single-cloud settings. As shown in Fig. 4, the architecture involves several entities with distinct roles.

In the setup phase (Step 1), the Certified Authority (CA) generates cryptographic keys: (pk_0, sk_0) for the DO, (pk_u^i, sk_u^i) for each QU, and a split strong key (SK_1, SK_2), which are respectively assigned to the two cloud servers (CSS and CCS). All public keys are shared among the entities. The DO then prepares the dataset. For sensitive data (Step 2), it preprocesses VD into PD, encrypts PD with DT-PKC to obtain $\widehat{PD}$, and uploads it to CSS. For less sensitive data (Step 3), it builds an R-tree index $Tree_R$, encrypts it with CIPE$_s$, and distributes the encrypted index $\widehat{Tree}_R$ to ESs for efficient query filtering.

When a QU issues a query (Step 4), it constructs $SQ = (\widehat{Rect}_q, E_{pk_0}(q), k)$ and sends it to a nearby ES. The ES evaluates $\widehat{Rect}_q$ over $\widehat{Tree}_R$, filters candidate results IR, and forwards them together with $(E_{pk_0}(q), k)$ to CSS (Step 5). Next, CSS and CCS jointly execute secure protocols (Step 6), and return the final result set $Result$ along with a verification object VO to the QU (Step 7). The QU then verifies the correctness of the result before finalizing the query.

4.2. Security model

DESMkNN is designed to address three security threats. First, CS cannot be fully trusted and may tamper with query results. Second, CS may act as honest-but-curious adversaries that attempt to infer sensitive information from the encrypted data. Third, QUs themselves may be curious and try to learn the query information of others.

To counter the risk of result tampering, DESMkNN incorporates a verification mechanism that ensures both correctness and completeness [27]. Correctness requires that every returned point $p \in Result$ remains unmodified and originates from the authentic database, while completeness guarantees that all true kNN results are included and no irrelevant points are omitted.

The other two threats are addressed by designing a secure index and a set of novel secure protocols that jointly preserve multiple dimensions of privacy [4,28]. Specifically, data privacy ensures that the database D remains hidden from the CS; query privacy requires that the content of a QU's query SQ is concealed from both the CS and other QUs; result privacy guarantees that only the QU can access the returned $Result$; and access-pattern privacy prevents the CS from learning which database entries satisfy a given query.

It is noteworthy that during system setup stage, CCS is prevented from compromising or collaborating with CSS. Furthermore, collusion between CS and QUs must be prevented throughout the query process.

5. DESMkNN construction

This section first introduces an optimized two-stage search framework that supports efficient and secure multi-user kNN queries with dynamic POIs updating. Subsequently, several well-designed secure protocols are proposed to enable private kNN search operations on the two-stage search framework.

5.1. Two-stage search framework

DESMkNN adopts a two-stage search framework, which consists of an initial filtering stage based on hierarchical clustering to effectively constrain the search range, followed by a precise search stage to achieve efficient querying.

Initial Filtering Stage: DO first preprocesses the dataset by using hierarchical clustering to construct a suitable $Tree_R$. Each node in the tree is encrypted by using the CIPE$_s$.EncI algorithm to ensure security. The $\widehat{Tree}_R$ is then uploaded to ESs. When a QU at position (x_q, y_q) initiates a query, they define a scope L and construct a rectangle $Rect_q$ centered at (x_q, y_q) with edge length L. Each dimension of $Rect_q$ is encrypted by using the CIPE$_s$.EncQ algorithm and sent to the nearby ES. The ES evaluates $\widehat{Rect}_q$ over $\widehat{Tree}_R$ to generate IR, which efficiently narrows down the candidate objects.

Precise Search Stage: Once receiving $(E_{pk_0}(q), k)$ and IR from ES, the dual-cloud servers collaboratively execute secure protocols over the preprocessed dataset to obtain the exact k nearest neighbors ($Result$). The servers also generate a verification object (VO) and send it with the $Result$ back to QU for checking. This stage ensures both accuracy and security of the kNN search.

5.2. Data pre-processing

To support DESMkNN, DO preprocesses the dataset before outsourcing, which aims to protect sensitive information while retaining the structural relationships required for queries. First, DO constructs a

Voronoi diagram VD from the dataset D, and encrypts the coordinates of each POI and query point q using DT-PKC. For every POI $p_i \in VD$, a unique label $\mathcal{L}_i = H(x_i|y_i)$ is generated through the SHA-256 hash function, which serves as a compact identifier. Subsequently, DO obtains the neighborhood $VN(p_i)$ and its corresponding label set $VN\mathcal{L}(p_i)$, then employs DT-PKC to encrypt the packaged $VN(p_i)$ after applying data packaging technology [29]. This technique helps handle multiple values together, which makes encryption more straightforward. To guarantee integrity, a signature $SIGp_i = H(H(p_i)|H(VN(p_i)))$ is created, where $H(VN(p_i))$ is obtained by hashing all neighbors together as

$$H(VN(p_i)) = H(H(p_{VN_1})|H(p_{VN_2})|...|H(p_{VN_{max}})).$$

Intuitively, this signature ensures any tampering with p_i or its neighbors can be detected. Since homomorphic encryption requires uniform input length, DO also performs incremental obfuscation: if a POI has fewer neighbors than the maximum in VD, dummy neighbors are added to conceal the actual degree. Afterward, each POI is represented by a sextuple

$$(E_{pk_0}(id), E_{pk_0}(p_i), E_{pk_0}(VN(p_i)), \mathcal{L}i, VN\mathcal{L}(p_i), SIGp_i),$$

which combines encrypted attributes, hashed labels, and a verifiable signature.

To further protect access pattern privacy, DO divides the sextuple table into buckets [8,9] of size w, which ensures queries operate over fixed-size groups instead of revealing individual record access. Since the final bucket may not be completely filled, DO pads it with randomly generated dummy records, which prevents inference attacks [30,31] where an adversary could deduce whether two queries target the same bucket based on its record count. At this point, DO completes preprocessing and securely outsources the bucketized sextuples to CSS.

5.3. Secure Square Distance Computation(SSDC)

The goal of SSDC is to compute the secure squared distance without revealing any valid coordinate information to CSS and CCS. The process is shown in Algorithm 1.

Initially, CSS randomly chooses 4 random numbers $r_1, r_2, r_4, r_5 \in \mathbb{Z}_N$, and chooses the functionality $F \in \{0,1\}$ (line 1–2). If $F = 1$, CSS calculates the encrypted coordinate differences $E_{pk_0}(A), E_{pk_0}(B)$ (line 3–5). If $F = 0$, the procedure is the same except that the positions of x_1 and x_2, as well as y_1 and y_2, are swapped when computing the differences (line 6–7). To mask these values and avoid direct leakage, CSS applies randomization with r_1 and r_2 (line 8). Subsequently, CSS partially decrypts the masked values a'', b'' by using the PSDec1 function to get a', b'(line 9). Eventually, CSS sends a'', b'', a', b' and $E_{pk_0}(A), E_{pk_0}(B)$ to CCS (line 10).

Upon receiving a series of encrypted values from CSS, CCS chooses a random number $r_3 \in \mathbb{Z}_N$ and decrypts the encrypted values to obtain a and b (line 11–12). To conceal the sign information of the differences, CCS applies a randomized comparison procedure (line 13–18). Specifically, depending on the outcomes of a versus 0 and related conditions, CCS produces three possible cases and outputs E_1 accordingly; this design prevents CSS from learning whether $x_1 - x_2$ or $y_1 - y_2$ is positive or negative. The same process is repeated for b to obtain E_2 (line 19). Finally, CCS returns E_1, E_2 to CSS (line 20).

Upon receiving a series of encrypted values from CCS, CSS further randomizes E_1 and E_2 with r_4 and r_5, then partially decrypts them to produce (c'', d'') and (c', d'), and sends these values to CCS (line 21–24). CCS completes the decryption (line 25), squares the plaintexts to derive $s = c^2$ and $z = d^2$ (line 26–27), and sends back $E_{pk_0}(s), E_{pk_0}(z)$ (line 28). Finally, CSS combines these ciphertexts through homomorphic operations to obtain D_1 and D_2, and computes the secure squared distance as $Distance = D_1 * D_2$.

Algorithm 1 Secure Squared Distance Computation

Require: CSS has $E_{pk_0}(x_1), E_{pk_0}(y_1), E_{pk_0}(x_2), E_{pk_0}(y_2)$; CSS has SK_1, pk_0; CCS has SK_2, pk_0;
Ensure: $E_{pk_0}(|x_1 - x_2|^2 + |y_1 - y_2|^2)$;
// Calculation in CSS:
1: Choose 4 random numbers $r_1, r_2, r_4, r_5 \in \mathbb{Z}_N$;
2: Randomly choose the functionality $F \in \{0,1\}$;
3: **if** $F = 1$ **then**
4: $E_{pk_0}(A) \leftarrow E_{pk_0}(x_1) * E_{pk_0}(x_2)^{N-1}$;
5: $E_{pk_0}(B) \leftarrow E_{pk_0}(y_1) * E_{pk_0}(y_2)^{N-1}$;
6: **else if** $F = 0$ **then**
7: Swap x_1 with x_2 and y_1 with y_2;
8: $a'' \leftarrow E_{pk_0}(A)^{r_1}$, $b'' \leftarrow E_{pk_0}(B)^{r_2}$;
9: $a' \leftarrow PSDec1(SK_1, a'')$, $b' \leftarrow PSDec1(SK_1, b'')$;
10: Send a'', b'', a', b' and $E_{pk_0}(A), E_{pk_0}(B)$ to CCS;
// Calculation in CCS:
11: Choose a random number $r_3 \in \mathbb{Z}_N$;
12: $a \leftarrow PSDec2(SK_2, a'', a')$, $b \leftarrow PSDec2(SK_2, b'', b')$;
13: **if** $a > 0$ **then**
14: $E_1 \leftarrow E_{pk_0}(A)$;
15: **else if** $E_{pk_0}(r_3) * E_{pk_0}(A)^{N-1} = E_{pk_0}(r_3)$ **then**
16: $E_1 \leftarrow E_{pk_0}(r_3)^0$;
17: **else**
18: $E_1 \leftarrow E_{pk_0}(A)^{N-1}$;
19: Apply the same steps to b to obtain E_2;
20: Send E_1, E_2 to CSS;
// Calculation in CSS:
21: $c'' \leftarrow E_1 * E_{pk_0}(r_4)$;
22: $c' \leftarrow PSDec1(SK_1, c'')$;
23: Apply the same steps to E_2, r_5 to obtain d'', d';
24: Send c'', c', d'', d' to CCS;
// Calculation in CCS:
25: $c \leftarrow PSDec2(SK_2, c'', c')$;
26: $s \leftarrow c * c$;
27: Apply the same steps to d'', d' to obtain d, z;
28: Send $E_{pk_0}(s), E_{pk_0}(z)$ to CSS;
// Calculation in CSS:
29: $D_1 \leftarrow E_{pk_0}(s) * E_1^{N-r_4} * E_1^{N-r_4} * E_{pk_0}(r_4 * r_4)^{N-1}$;
30: $D_2 \leftarrow E_{pk_0}(d) * E_2^{N-r_5} * E_2^{N-r_5} * E_{pk_0}(r_5 * r_5)^{N-1}$;
31: $Distance \leftarrow E_{pk_0}(|x_1 - x_2|^2 + |y_1 - y_2|^2) \leftarrow D_1 * D_2$;

5.4. Secure Minimum Computation(SMC)

The goal of SMC is to compare two secure squared distances obtained by SSDC, determine the smaller one, and also obtain the corresponding id_{min} and $\mathcal{L}_{min}$. The process is shown in Algorithm 2.

To start with, CSS generates 7 random numbers and randomly selects a functionality F, in a manner similar to SSDC (line 1–2). If $F = 1$, CSS masks the differences between the distances, identifiers, and location labels by incorporating random numbers either as multiplicative factors or as exponents (line 3–10). For example, the key step

$$E_{pk_0}(\alpha) \leftarrow (E_{pk_0}(d_1) * E_{pk_0}(d_2)^{N-1})^{r_\alpha}$$

ensures that CCS cannot infer the exact magnitude of d_1 and d_2 with no less than 1/2 probability, which enables to preserve the magnitude relationship with semantic security. If $F = 0$, the roles of d_1 and d_2 are swapped, and the same randomization procedure follows (line 11–12). After randomization, CSS partially decrypts one of the masked values to obtain α_1 and sends it together with the corresponding encrypted terms to CCS (line 13–14).

Upon receiving these values, CCS decrypts α_1 to obtain α_2 (line 15). By checking whether the bit-length of α_2 exceeds half modulus size, CCS

decides whether d_1 or d_2 is smaller, and records this decision in a flag w (line 16–19). Using w and the remaining encrypted values from CSS, CCS computes three encrypted auxiliary terms that encode the correct selection of the minimum distance, identifier, and label (line 20–22). These results, along with w, are then sent back to CSS (line 23).

Algorithm 2 Secure Minimum Computation

Require: CSS has $E_{pk_0}(d_1), E_{pk_0}(d_2), E_{pk_0}(id_1), E_{pk_0}(id_2), E_{pk_0}(\mathcal{L}_1), E_{pk_0}(\mathcal{L}_2)$;
CSS has SK_1, pk_0; CCS has SK_2, pk_0;
Ensure: $E_{pk_0}(d_{min}), E_{pk_0}(id_{min}), E_{pk_0}(\mathcal{L}_{min})$;
// Calculation in CSS:
1: Choose 7 random numbers $r_\alpha, r_\beta, r_\gamma, r_\delta, r_\epsilon, r_\zeta, r_\eta \in \mathbb{Z}_N$;
2: Randomly choose the functionality $F \in \{0, 1\}$;
3: **if** $F = 1$ **then**
4: $E_{pk_0}(\alpha) \leftarrow (E_{pk_0}(d_1) * E_{pk_0}(d_2)^{N-1})^{r_\alpha}$;
5: $E_{pk_0}(\beta) \leftarrow (E_{pk_0}(d_1) * E_{pk_0}(d_2)^{N-1} * E_{pk_0}(r_\beta))$;
6: $E_{pk_0}(\gamma) \leftarrow (E_{pk_0}(d_2) * E_{pk_0}(d_1)^{N-1} * E_{pk_0}(r_\gamma))$;
7: $E_{pk_0}(\delta) \leftarrow (E_{pk_0}(id_1) * E_{pk_0}(id_2)^{N-1} * E_{pk_0}(r_\delta))$;
8: $E_{pk_0}(\epsilon) \leftarrow (E_{pk_0}(id_2) * E_{pk_0}(id_1)^{N-1} * E_{pk_0}(r_\epsilon))$;
9: $E_{pk_0}(\zeta) \leftarrow (E_{pk_0}(\mathcal{L}_1) * E_{pk_0}(\mathcal{L}_2)^{N-1} * E_{pk_0}(r_\zeta))$;
10: $E_{pk_0}(\eta) \leftarrow (E_{pk_0}(\mathcal{L}_2) * E_{pk_0}(\mathcal{L}_1)^{N-1} * E_{pk_0}(r_\eta))$;
11: **else if** $F = 0$ **then**
12: Swaps the roles of d_1, id_1, $\mathcal{L}_1$ with d_2, id_2, $\mathcal{L}_2$.
13: $\alpha_1 \leftarrow PSDec1(SK_1, E_{pk_0}(\alpha))$;
14: Send $\alpha_1, E_{pk_0}(\alpha), E_{pk_0}(\beta), E_{pk_0}(\gamma), E_{pk_0}(\delta), E_{pk_0}(\epsilon), E_{pk_0}(\zeta), E_{pk_0}(\eta)$ to CSS;
// Calculation in CCS:
15: $\alpha_2 \leftarrow PSDec2(SK_2, E_{pk_0}(\alpha), \alpha_1)$;
16: **if** $Length(\alpha_2) > Length(N)/2$ **then**
17: $w \leftarrow 1$;
18: **else**
19: $w \leftarrow 0$;
20: $E_{pk_0}(\theta) \leftarrow (E_{pk_0}(\beta)^{1-w} * E_{pk_0}(\gamma)^w)^{N-1}$;
21: $E_{pk_0}(\vartheta) \leftarrow (E_{pk_0}(\delta)^{1-w} * E_{pk_0}(\epsilon)^w)^{N-1}$;
22: $E_{pk_0}(\iota) \leftarrow (E_{pk_0}(\zeta)^{1-w} * E_{pk_0}(\eta)^w)^{N-1}$;
23: Send $w, E_{pk_0}(\theta), E_{pk_0}(\vartheta), E_{pk_0}(\iota)$ to CSS;
// Calculation in CSS:
24: **if** $s = w$ **then**
25: $E_{pk_0}(d_{min}) = E_{pk_0}(d_2) * E_{pk_0}(\theta) * E_{pk_0}(w)^{r_\gamma} * (E_{pk_0}(1-w))^{r_\beta}$;
26: $E_{pk_0}(id_{min}) = E_{pk_0}(id_2) * E_{pk_0}(\vartheta) * E_{pk_0}(w)^{r_\epsilon} * (E_{pk_0}(1-w))^{r_\delta}$;
27: $E_{pk_0}(\mathcal{L}_{min}) = E_{pk_0}(\mathcal{L}_2) * E_{pk_0}(\iota) * E_{pk_0}(w)^{r_\eta} * (E_{pk_0}(1-w))^{r_\zeta}$;
28: **else**
29: Swaps the roles of d_2, id_2, $\mathcal{L}_2$ with d_1, id_1, $\mathcal{L}_1$.

At the end of Algorithm 2, CSS computes 3 encrypted values: $E_{pk_0}(d_{min}), E_{pk_0}(id_{min}), E_{pk_0}(\mathcal{L}_{min})$ via homomorphic encryption. The computation applies to $s = w$ and $s \neq w$ (line 24-29). In this way, the protocol securely determines the minimum distance and its associated information without revealing any intermediate values.

5.5. Secure Set Difference(SSD)

The goal of SSD is to securely compute the set difference between two encrypted sets, which allows CSS to obtain the elements in S_1 that are not in S_2, without exposing any plaintext values. To achieve this, CSS holds the encrypted sets $\widehat{S_1}$ and $\widehat{S_2}$ together with SK_1, while CCS holds SK_2. The protocol begins with CSS initializing an empty table and iteratively processing each encrypted element in $\widehat{S_1}$ (line 1–2). For each comparison with an element in $\widehat{S_2}$, CSS generates a random blinding factor and constructs a masked comparison token that conceals the difference between the two values (line 3–6). This token is then partially decrypted using SK_1, producing an auxiliary value that, together with the token, is stored in a permuted list under a pseudo-random permutation to prevent linkability (line 7–9). After completing all comparisons, CSS sends the resulting table to CCS for further processing (line 10).

On the CCS side, the server initializes an empty set and parses the received tokens (line 11–12). Each token is decrypted with SK_2, and whenever a decryption reveals equality between an element of $\widehat{S_1}$ and $\widehat{S_2}$, the corresponding index is added to the set (line 13–15). This set, containing the indices of overlapping elements, is then returned to CSS (line 16). Finally, CSS uses the inverse permutation to locate the original positions and removes the identified elements from $\widehat{S_1}$ (line 17–19). The remaining encrypted elements constitute the secure set difference $\widehat{S}'$, which represents all values in S_1 but not in S_2 (line 20).

Algorithm 3 Secure Set Difference

Require: CSS has two sets of encrypted values
$\widehat{S_1} = \{E_{pk_0}(x_1), ..., E_{pk_0}(x_M)\}$;
$\widehat{S_2} = \{E_{pk_0}(y_1), ..., E_{pk_0}(y_T)\}$;
CSS has SK_1; CCS has SK_2;
Ensure: CSS obtains an encrypted difference set $\widehat{S}'$;
// Calculation in CSS:
1: Initialize T to an empty table;
2: **for** the i−th element $E_{pk_0}(x_i) \in \widehat{S_1}$ **do**
3: Initialize t to an empty list;
4: **for** all $E_{pk_0}(y_j) \in \widehat{S_2}$ in random order **do**
5: Generate a random number $r_{i,j}$;
6: $t_{i,j}[0] \leftarrow (E_{pk_0}(x_i) * E_{pk_0}(y_j)^{N-1})^{r_{i,j}}$;
7: $t_{i,j}[1] \leftarrow PSDec1(SK_1, t_{i,j}[0])$;
8: Append $t_{i,j}$ to t;
9: $T[\pi(i)] \leftarrow t$;
10: Send T to CCS;
// Calculation in CCS:
11: Initialize V to an empty set;
12: **for** $i \in [M]$ **do**
13: Parse $T[i]$ as $(t_{i,1}, ..., t_{i,T})$;
14: **if** $\exists t_{i,j} \in T[i] \cap PSDec2(SK_2, t_{i,j}[0], t_{i,j}[1])$ **then**
15: Add i into set V;
16: Send V to CSS;
// Calculation in CSS:
17: **for** each element i in V **do**
18: $j \leftarrow \pi^{-1}(i)$;
19: Remove the j−th element $E_{pk_0}(x_j)$ from $\widehat{S_1}$;
20: $\widehat{S}' \leftarrow \widehat{S_1}$;

5.6. Secure Insertion(SI)

To support secure data insertion in databases, DESMkNN innovatively proposes a secure insertion protocol. When DO inserts a new POI into the database, two key problems must be addressed.

- How to determine the insertion position of the POI?
- How to update $Tree_R$ and VD?

The first problem can be effectively resolved by CIPE$_s$. First, DO generates an insertion query rectangle $Rect_{ins}$ for the POI to be inserted, similar to generating a query rectangle $Rect_q$ for the query point q in the initial filtering stage, where the L of the rectangle can be customized. Then, DO encrypts each dimension of $Rect_{ins}$ with CIPE$_s$.EncQ algorithm and sends $\widehat{Rect_{ins}}$ to ES near the inserted POI. ES will evaluate the obtained $\widehat{Rect_{ins}}$ over $\widehat{Tree_R}$ to obtain the insertion position.

Once the insertion position is determined, the label of the inserted POI can be added to the $Tree_R$, thus completing the update of $Tree_R$. To address the problem of how to update VD, the Bowyer-Watson algorithm [32,33] is introduced. The Bowyer-Watson algorithm is an

incremental method that updates VD by progressively updating the Delaunay triangulation. When inserting a new point, algorithm first identifies all the affected triangles, then removes them and reconstructs the triangulation mesh by using the new point and the boundary of the cavity, which ensures that the new Delaunay triangulation is valid. Since VD and Delaunay triangulation are duals, when the Delaunay triangulation is updated by using the Bowyer-Watson algorithm, VD is updated accordingly. When a new generating point is inserted, the shape and boundaries of the Voronoi cells are adjusted. Therefore, DO obtains the updated Voronoi diagram based on the Bowyer-Watson algorithm and can obtain the encrypted id of the newly inserted POI: $E_{pk_0}(id_{ins})$, the encrypted inserted POI: $E_{pk_0}(p_{ins})$, the label of the newly inserted POI: $\mathcal{L}_{ins}$, the encrypted Voronoi neighbors: $E_{pk_0}(VN(p_{ins}))$, the encrypted labels of Voronoi neighbors: $E_{pk_0}(VN\mathcal{L}(p_{ins}))$, and the signature: SIG_{ins} used for verification. Finally, these six values are organized into a tuple and sent to CSS for storage. As shown in Fig. 5, the secure insertion in the R-tree is highlighted with green lines.

Algorithm 4 Secure kNN Query

Require: CSS has $IR, E_{pk_0}(q), SK_1$;
CCS has SK_2;
Ensure: CSS obtains the encrypted search result $Result$;
// Calculations in CSS and CCS:
1: CSS initializes R, C, De to empty sets;
2: **for** each triple $(E_{pk_0}(p_i), E_{pk_0}(id_i), E_{pk_0}(\mathcal{L}_i)) \in IR$ **do**
3: CSS appends $E_{pk_0}(P_i)$ to C;
4: CSS with input $(C, E_{pk_0}(q), SK_1, pk_0)$ and CCS with input (SK_2, pk_0) run SSDC protocol, and CSS obtains $\{Distance_1, ..., Distance_{|C|}\}$;
5: **if** $|C| \geq k$ **then**
6: $(\{Distance_i, E_{pk_0}(id_i), E_{pk_0}(\mathcal{L}_i)\}_{i=1}^{|C|}, SK_1, pk_0)$ as input in CSS and CCS with input (SK_2, pk_0) run SMC protocol, and CSS puts $(E_{pk_0}(id_{i^*}))_{i=1}^{k}$ into $Result$;
7: **else**
8: $(\{Distance_i, E_{pk_0}(id_i), E_{pk_0}(\mathcal{L}_i)\}_{i=1}^{|C|}, SK_1, pk_0)$ as input in CSS and CCS with input (SK_2, pk_0) run SMC protocol, and CSS puts $(E_{pk_0}(id_1^*))$ into $Result$ and puts $(E_{pk_0}(\mathcal{L}_1^*))$ into De;
9: CSS and CCS collaborate to run SCR protocol to get the row corresponding to the $E_{pk_0}(id_1^*)$;
10: CSS with input $(E_{pk_0}(VN\mathcal{L}(p_1^*)), De, SK_1)$ and CCS with input SK_2 run SSD protocol, and CSS obtains $VN\mathcal{L}'(p_1^*)$;
11: **for** $E_{pk_0}(p_j) \in E_{pk_0}(VN(p_1^*)) \cap VN\mathcal{L}'(p_1^*)$ **do**
12: CSS puts $E_{pk_0}(p_j)$ into C and $E_{pk_0}(\mathcal{L}_j)$ into De;
13: CSS and CCS collaborate to run SSD and SMC protocols to select the POI closest to q from C again, and removing it from C;
14: CSS inserts $E_{pk_0}(id_2^*)$ into $Result$;
15: **while** $|R| < k$
16: Repeat line 9-14;

5.7. Secure Deletion(SD)

To support secure data deletion in database, DESMkNN innovatively proposes a secure deletion protocol. First, DO generates an deletion query rectangle $Rect_{del}$ for the POI to be deleted, where the L of the rectangle can be customized. Then, DO encrypts each dimension of $Rect_{del}$ with the CIPE$_s$.EncQ algorithm and sends $\widehat{Rect_{del}}$ to ES near the deleted POI. ES will evaluate the obtained $\widehat{Rect_{del}}$ over $\widehat{Tree_R}$ to obtain the deletion position.

Once the deletion position is determined, DO sends $\mathcal{L}_{del}$, which is the label of the POI, to ES near the deleted POI. ES deletes the POI label from the data at deletion location based on $\mathcal{L}_{del}$ sent by DO. At this point, the deletion update of $Tree_R$ is completed.

Similar to SI protocol, DESMkNN introduces a Delaunay triangulation-based dynamic deletion and update algorithm for Voronoi diagrams. The key idea behind dynamic deletion and update algorithm is that Voronoi diagrams and Delaunay triangulations are dual to each other: the vertices of Delaunay triangles correspond to the vertices of Voronoi diagram, and the edges of Delaunay triangles correspond to the edges of Voronoi diagram. The Delaunay triangulation-based Voronoi diagram dynamic deletion and update algorithm leverages the duality of Delaunay triangles to efficiently update Voronoi diagram. When a point is deleted, the corresponding Delaunay triangles are removed, and the algorithm updates the connectivity of affected neighboring triangles to maintain the Delaunay condition, which ensures that the triangulation is reconstructed. Then, based on the new Delaunay triangulation, Voronoi diagram's boundaries are updated to ensure the correct topological structure of the diagram.

Similarly, DO obtains the updated VD and the labels of affected POIs $\mathcal{L}_{affect_i}$, the encrypted Voronoi neighbors $E_{pk_0}(VN(p_{affect_i}))$, the encrypted labels of Voronoi neighbors $E_{pk_0}(VN\mathcal{L}(p_{affect_i}))$, and the signature SIG_{affect_i} used for verification. Finally, these four values are organized into a quadruple and sent to CSS, which updates the database based on the labels of the affected POIs. As shown in Fig. 5, the secure deletion in the R-tree is highlighted with red lines.

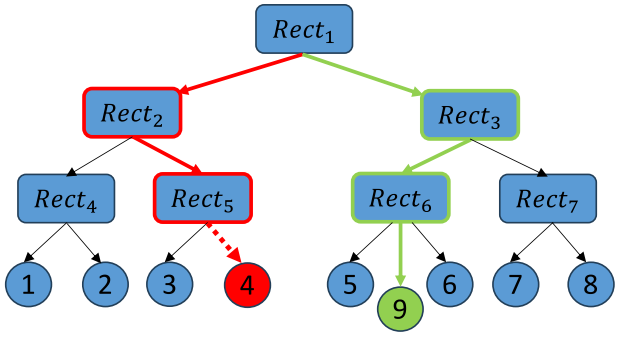


Fig. 5. Secure insertion and deletion in R-tree. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 5 Secure Transformation

Require: CSS has $E_{pk_0}(a), SK_1$;
CCS has SK_2;
Ensure: CSS obtains $E_{pk_u}(a)$;
// Calculations in CSS:
1: Choose one random number $r \in \mathbb{Z}_N$;
2: $E_{pk_0}(\alpha) = E_{pk_0}(a) * E_{pk_0}(r)$;
3: $\alpha' \leftarrow PSDec1(SK_1, E_{pk_0}(\alpha))$;
4: Send $E_{pk_0}(\alpha), \alpha'$ to CCS;
// Calculations in CCS:
5: $\alpha \leftarrow PSDec2(SK_2, E_{pk_0}(\alpha), \alpha')$;
6: Send $E_{pk_u}(\alpha)$ to CSS;
// Calculations in CSS:
7: $E_{pk_u}(a) = E_{pk_u}(\alpha) * E_{pk_u}(r)^{N-1}$;

6. DESMkNN query processing

This section provides a detailed introduction to DESMkNN query processing, which consists of two parts: secure kNN query processing and verification processing.

6.1. Secure kNN query processing

Based on comprehensive search framework, DESMkNN proposes a secure and verifiable query processing strategy, which is divided into three steps as follows:

- *Step 1. Calculating k nearest neighbors*: The specific details and procedures are illustrated in Algorithm 4. First, CSS will create three new sets, which includes the result set $Result$, the candidate set C, and the deduplication set De (line 1). After initial filtering stage, CSS has $IR = \{(E_{pk_0}(p_i), E_{pk_0}(id_i), E_{pk_0}(\mathcal{L}_i))\}$. Next, CSS will insert each encrypted POI $E_{pk_0}(p_i)$ from IR into C (line 2–3). Since CSS has already stored the encrypted query point $E_{pk_0}(q)$, the SSDC protocol is executed for each intermediate POI to obtain the secure squared distance between each POI and the query point (line 4). If $|C| \geq k$, which means that the required k POIs can be found in IR, CSS and CCS will collaborate to execute SMC protocol to obtain the desired k POIs (line 5–6). If $|C| < k$, CSS and CCS collaborate to execute the SMC protocol to obtain the nearest POI, and insert the corresponding $E_{pk_0}(id_1^*)$ into $Result$, and the corresponding $E_{pk_0}(\mathcal{L}_1^*)$ into De (line 7–8). To further get the next nearest neighbor, CSS and CCS collaborate to execute the SCR protocol [8,9], to get the row corresponding to the $E_{pk_0}(id_1^*)$: $E_{pk_0}(VN(p_1^*)), VN\mathcal{L}(p_1^*), SIG_{p_1}^*$ (line 9). CSS and CCS collaborate to execute the SSD protocol, with two input sets $VN\mathcal{L}(p_1^*)$ and De. CSS obtains $VN\mathcal{L}'(p_1^*)$ (line 10). If one POI $E_{pk_0}(P_j)$ in $E_{pk_0}(VN(p_1^*))$ also exists in $VN\mathcal{L}'(p_1^*)$, $E_{pk_0}(p_j)$ is added to C, and $E_{pk_0}(\mathcal{L}_j)$ is added to De (line 11–12). CSS and CCS collaborate to execute SSD protocol and SMC protocol, which selects the POI closest to the query point from C again and removes it from C (line 13). CSS inserts $E_{pk_0}(id_2^*)$, which corresponds to the obtained point, into $Result$ and checks whether the content in $Result$ meets the requirements of kNN queries. If not, SkQ will repeat line 9–14.

- *Step 2. Generating verification object*: During secure kNN queries, DESMkNN also need to generate VO. By collaborating to execute the SCR protocol, CSS and CCS can obtain $E_{pk_0}(VN(p_i))$ and SIG_{p_i} from the row, which corresponds to p_i. Additionally, algorithm 5 enables key conversion, which transforms $E_{pk_0}(VN(p_i))$ into $E_{pk_u}(VN(p_i))$. At last, CSS adds $E_{pk_u}(VN(p_i))$ and $E_{pk_u}(SIG_{p_i})$ of each result point into VO.

- *Step 3. Returning results and verification object to QU*: Based on secure protocols we proposed, CSS can directly retrieve the final results encrypted with pk_u in order, without needing an additional transformation process. Therefore, CSS puts the final points into $Result$ and sends it, along with VO, to QU.

6.2. Verification processing

QU utilizes $Result$ and VO to authenticate the correctness and completeness of $Result$.

- *Verifying correctness*: Recall the definition of correctness described in the security model, which means that each returned point $p \in Result$ remains unmodified and is an authentic entry in the original database. To verify the correctness of $Result$, QU first decrypts VO by using his private key sk_u to obtain $\{VN(p_i), SIG_{p_i}\}$. Next, QU uses the obtained $VN(p_i)$ to compute $H(VN(p_i))$ and further calculates $H(H(p_i)|H(VN(p_i)))$ (the specific method has been detailed in Data Pre-processing). Finally, QU only needs to check whether SIG_{p_i} matches the computed $H(H(p_i)|H(VN(p_i)))$ to verify correctness.

- *Verifying completeness*: Similar to correctness, completeness is defined as follows: all the points returned are valid solutions to the kNN query, while the points not returned do not correspond to the actual answers. First, assume that p_i^* represents the ith nearest point to the query point q in $Result$. Subsequently, based on the properties of the Voronoi diagram, $VC(p_i^*)$ can be derived from $VN(p_i^*)$ and p_i^*. The specific process is divided into four steps: (1) determine the coordinates of the neighboring points; (2) calculate the perpendicular bisectors between p_i^* and each neighboring point; (3) identify the intersection points of all these perpendicular bisectors, these intersection points form the vertices of the polygon, which represent the Voronoi cell; (4) connect these vertices in either a clockwise or counterclockwise order to form the Voronoi cell surrounding the point p_i^*. Thereafter, the final verification is conducted based on the two important properties of the Voronoi diagram. The first step is to determine whether q lies within $VC(p_1^*)$. If it does, p_1^* is confirmed as the nearest POI; otherwise, the verification process is terminated immediately. The second step is to test each point (except for p_1^*) in $Result$ individually, which determines whether $p_i^* \in \{VN(p_1^*) \cup \cdots \cup VN(p_{i-1}^*)\}, i > 1$. If it does, p_i^* is confirmed as the ith nearest POI.

7. Analysis

7.1. Computational complexity

To verify the efficiency of DESMkNN, we analyze the computational complexity of all four entities involved in the system: DO, QU, ESs, and dual-cloud servers. Let e_c and d_c denote the encryption and decryption operations of CIPE$_S$, and let e_{dt} and d_{dt} represent the encryption and decryption operations of DT-PKC.

(1) *DO*: In the data pre-processing stage, DO needs to generate $Tree_R$ and VD based on the database D. $Tree_R$ and the PD generated from VD are encrypted by using CIPE$_S$ and DT-PKC, respectively. Therefore, the total computational complexity is

$$O(n)e_c + O(n * M)e_{dt},$$

where M represents the maximum number of neighbors in VD.

(2) *QU*: Due to the key conversion mechanism in Algorithm 5, QU only needs to perform a single DT-PKC decryption to obtain the final result and VO. Thus, the computational cost is $O(1)d_{dt}$.

(3) *ESs*: The ESs perform initial filtering by evaluating the encrypted query rectangle $\widehat{Rect_q}$ over the encrypted R-tree $\widehat{Tree_R}$ to generate the intermediate result set IR. Their total computational complexity is $O(log_n)d_c$.

(4) *Dual-Cloud Servers*: The dual-cloud servers undertake the precise search stage and therefore incur the highest computational complexity, as this stage requires executing several secure subprotocols. Specifically, the SSDC protocol is used to compute the secure squared distance between the query point q and each POI in the intermediate result set IR. The SMC protocol is responsible for comparing encrypted distance values and obtaining the corresponding encrypted identifiers and location records. To determine the nearest POI among candidates, the SMC protocol must be executed n-1 times. In addition, the SSD protocol computes the set difference between two encrypted sets and must perform DT-PKC decryption $|\widehat{S_1}| * |\widehat{S_2}|$ times. The overall complexity depends on whether the number of candidates in IR is greater than or smaller than k. When $|IR| > k$, the SkQ protocol repeatedly invokes the SMC protocol to iteratively determine the top-k POIs, which requires $(|IR|-1+|IR|-k) * k/2$ executions in total. In this case, the computational complexity of the precise search stage is

$$O(|IR| * k)(e_{dt} + d_{dt}),$$

Table 3
Computational complexity of existing approaches and DESMkNN.

	DO	QU	ES	Dual-cloud servers
DESMkNN	$O(n)e_c + O(n*M)e_{dt}$	$O(1)d_{dt}$	$O(log_n)d_c$	$\begin{cases} O(\|IR\|*k)(e_{dt}+d_{dt}) \\ O(\|IR\|+k^2*M)e_{dt}+O(\|IR\|+k*(\sqrt{n}+k*M))d_{dt} \end{cases}$
MSVkNN [9]	$O(m^2*g+n*M)e_{dt}$	$O(1)d_{dt}$	–	$O(k*(n+M))e_{dt}+O(k*(\sqrt{n}+M))d_{dt}$
SecVKQ [14]	$O(n)e_c + O(n*M)e_p$	$O(1)(e_c+e_p)$	$O(log_n)d_c$	$\begin{cases} O(\|IR\|*k)(e_p+d_p) \\ O(\|IR\|+k^2*M)(e_p+d_p) \end{cases}$
SVkNN [8]	$O(m^2*g+n*M)e_p$	$O(1)d_p$	–	$O(k*(n+M))e_p+O(k*(\sqrt{n}+M))d_p$

Notations: Let n represents the size of dataset D, k represents the search parameter for kNN search, and M represents the maximal number of Voronoi neighbors. m refers to the number of grids, while g represents the maximum number of grid points, as discussed in [8,9].

Table 4
Comparison of communication costs (MB) under the setting of $K = \{1024, 2048\}$.

n	DESMkNN				MSVkNN			
	California		San Francisco		California		San Francisco	
	1024	2048	1024	2048	1024	2048	1024	2048
1024	6.1	12.7	5.9	12.3	6.5	13.1	6.1	12.4
2048	12.8	27.8	11.9	25.6	14.3	31.4	13.9	30.7

When $|IR| < k$, the nearest POI is first identified by using $|IR|-1$ SMC comparisons. Next, the SCR protocol is executed to locate the bucket row containing this POI, after which the remaining $k-1$ POIs are obtained through the subsequent steps of SkQ. In this case, the computational complexity of the precise search stage is

$$O(|IR| + k^2 * M)e_{dt} + O(|IR| + k * (\sqrt{n} + k * M))d_{dt}.$$

where M denotes the maximum number of neighbors in the Voronoi diagram. The comparison results between DESMkNN and existing secure kNN query schemes are summarized in Table 3.

Moreover, The computational complexity of POI insertion and deletion in DESMkNN is $O(logn + log(M_1))$ on average, which is asymptotically equivalent to $O(log(M_1 n))$. Here, M_1 represents the number of neighboring POIs affected by the local Voronoi diagram update. This complexity arises from updating the encrypted R-tree and locally maintaining the Voronoi diagram.

7.2. Communication complexity

In this subsection, the communication cost incurred during the entire query processing is evaluated. As shown in Table 4, it presents the communication cost of DESMkNN compared with MSVkNN. It is observed that DESMkNN consistently incurs the lowest communication cost. These experimental results align well with the theoretical analysis.

7.3. Security analysis

To establish the security of the proposed subprotocol, it is important to highlight that the semantic security of the DT-PKC cryptosystem has been proven in [19]. Additionally, in accordance with the formal security definition of multiparty computation introduced in [29] and [34], the framework of the simulation paradigm proposed in [35] is adopted. Specifically, the simulation paradigm requires that the view of each participant in the protocol can be simulated based solely on its input and output, which ensures that no participant gains any additional information from the protocol. In other words, the real execution of each subprotocol is computationally indistinguishable from its simulated counterpart. For clarity, the SSDC and SMC are formally demonstrated as examples, and other protocols we proposed can be proven in a similar manner.

Theorem 1. *The DT-PKC cryptosystem described in Section 3 is semantically secure under the assumed intractability of the DDH problem over $\mathbb{Z}^*_{N^2}$. This ensures that ciphertexts produced by DT-PKC reveal no information about the underlying plaintexts, even to computationally bounded adversaries (The details of the proof can be referred to [19]).*

Theorem 2 (*Composition Theorem [35]*)**.** *If a protocol is composed of multiple subprotocols, each of which is secure under the simulation paradigm, and all intermediate values are either random or pseudorandom, the composed protocol is secure. This theorem allows the security of DESMkNN to be deduced from the security of its individual subprotocols.*

Theorem 3 (*Security of SSDC*)**.** *Assuming DT-PKC is semantically secure, the SSDC subprotocol securely computes encrypted squared distances between the query point and candidate points in IR for semi-honest adversaries.*

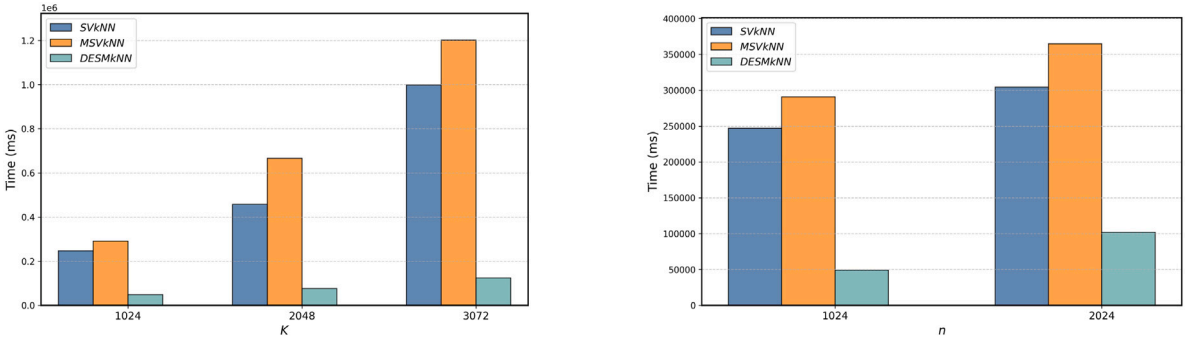
Proof. In SSDC, the cloud server's view consists of the ciphertexts a'', b'', a', b', which are derived from plaintext differences scaled by random factors, and the encrypted comparison results E_1, E_2. The simulated view $\prod^s_{CCS}(SSDC)$ is constructed by sampling all elements uniformly at random from the appropriate domain. The semantic security of DT-PKC ensures that a'', b'', a', b' are computationally indistinguishable from the corresponding simulated values $(a''_s, b''_s, a'_s, b'_s)$. Similarly, the randomized encryption of the comparison outcomes E_1, E_2 ensures that these values are indistinguishable from their simulated counterparts E_{1_s}, E_{2_s}. This demonstrates that the real execution reveals no additional information beyond what is contained in the input and output, which confirms the security of SSDC. For CSS, the execution image is $\prod_{CCS}(SSDC) = \{E_1, E_2\}$, and the simulated image is $\prod^s_{CCS}(SSDC) = \{E_{1_s}, E_{2_s}\}$. Since E_1, E_2 are produced by randomized procedures, they are computationally indistinguishable from E_{1_s}, E_{2_s}, which further supports the security argument.

Theorem 4 (*Security of SMC*)**.** *Assuming DT-PKC is semantically secure, the SMC protocol securely compares encrypted distance values and returns encrypted identifiers or labels.*

Proof. In SMC, the server's view contains ciphertexts $(E_{pk_0}(\alpha), \alpha_1, \alpha_2)$ and a local output bit w. The simulated view $\prod^s_{CCS}(SMC)$ is obtained by sampling all elements randomly. Semantic security guarantees that $(E_{pk_0}(\alpha), \alpha_1)$ are indistinguishable from their simulated counterparts $(E_{pk_0}(\alpha)_s, \alpha_{1_s})$. Additionally, α_2 is derived from random coin flips and is indistinguishable from α_{2_s}. The local output bit w also matches the distribution of the simulated w_s. Hence, the simulated view is computationally indistinguishable from the real view, which confirms the security of SMC.

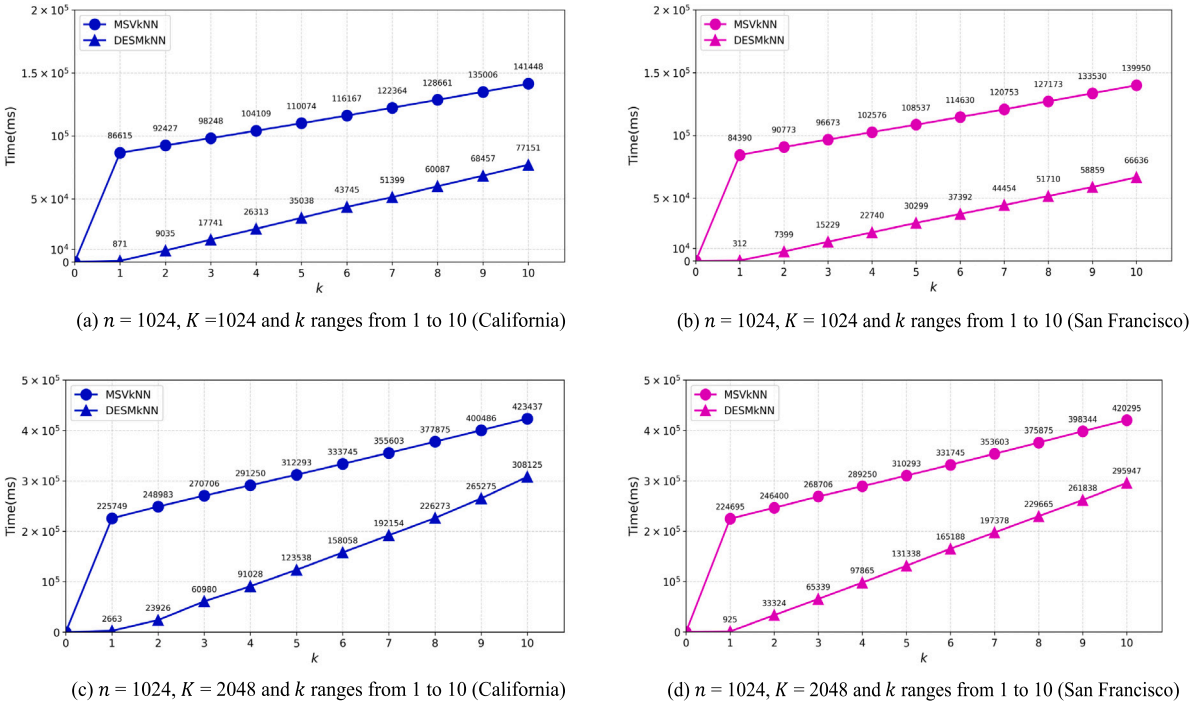
Theorem 5 (*Security of DESMkNN*)**.** *If DT-PKC is semantically secure, DESMkNN is secure under the semi-honest model.*

Proof. Since each subprotocol (SSDC, SMC, SSD, and others) produces views indistinguishable from their respective simulated views, and all



(a) n = 1024 and K ranges from 1024 to 3072 (b) K = 1024 and n ranges from 1024 to 2024

Fig. 6. The data processing time with varying parameters.



(a) n = 1024, K =1024 and k ranges from 1 to 10 (California) (b) n = 1024, K = 1024 and k ranges from 1 to 10 (San Francisco)

(c) n = 1024, K = 2048 and k ranges from 1 to 10 (California) (d) n = 1024, K = 2048 and k ranges from 1 to 10 (San Francisco)

Fig. 7. Comparison of search time between MSVkNN and DESMkNN on two datasets (k = 1 to 10).

intermediate values are either DT-PKC ciphertexts or explicitly randomized, the composition theorem applies. Consequently, the overall DESMkNN protocol is secure, ensuring confidentiality of the database, privacy of queries, and integrity of computation.

In DESMkNN, a quantitative security comparison across existing methods is not conducted due to significant differences in their threat models, cryptographic assumptions, and supported functionalities, which make such evaluation extremely difficult. Instead, DESMkNN focuses on formally achieving and proving multiple security properties that prior methods do not simultaneously provide. DESMkNN ensures data privacy, query privacy, result privacy, and access patterns privacy, while also supporting result verification, multi-user querying, and dynamic updates to the encrypted POIs database in outsourced POIs queries, which prior methods cannot achieve simultaneously.

8. Experimental evaluation

This section evaluates the computational cost of DESMkNN by using real-world datasets for spatial databases: California Road Network and San Francisco Road Network. A comparison is made between DESMkNN and scheme MSVkNN [9] in different phases.

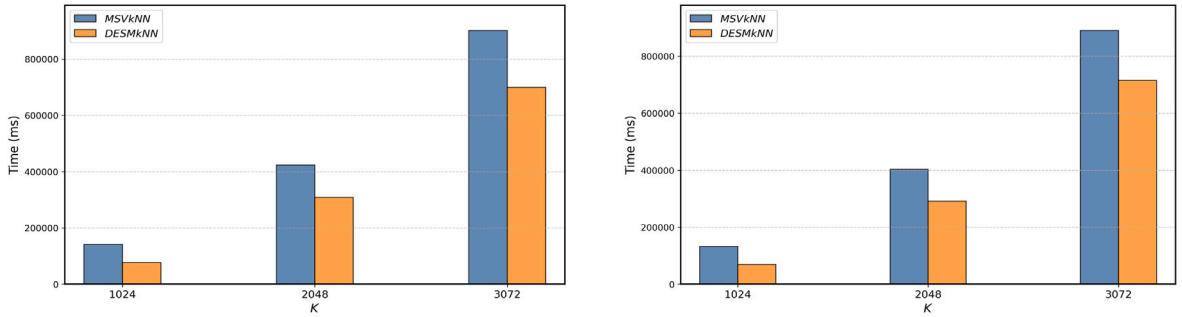
8.1. Parameter setting

The evaluation of DESMkNN is carried out on a system equipped with an Intel Core i7-14650HQ processor, clocked at 2.80 GHz, and 16 GB of RAM, which runs Windows 11. For this purpose, the DT-PKC cryptosystem is implemented by using the JAVA development kit, which forms the core element of the proposed protocol.

In the experiment, the dataset size n ranges from 1024 to 2024. The search parameter k is set between 1 and 10. The key size K of the DT-PKC cryptosystem are selected from $\{1024, 2048, 3072\}$. These settings apply to all values of n, k, K in the experiment. While implementing the MSVkNN and SVkNN schemes, the grid granularity is fixed at 90 and the cryptographic hash functions are implemented via HMAC-SHA-256.

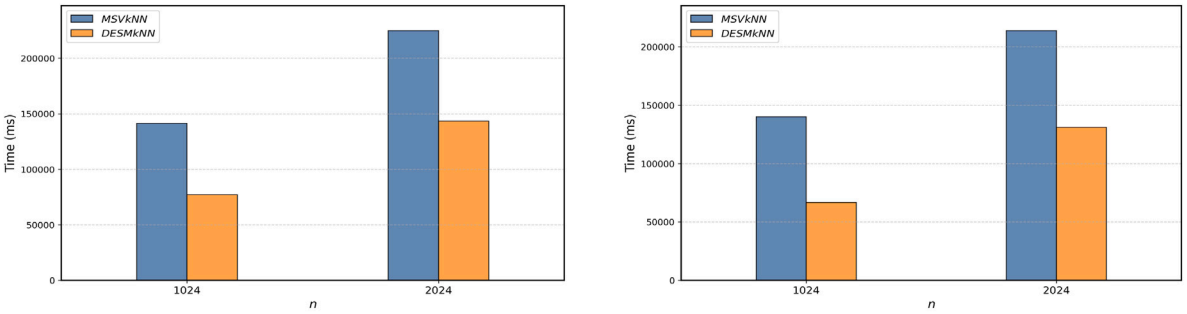
8.2. Experiment results

The following analysis of the experimental results will focus on DO and Dual-Cloud Servers. It should be noted that the experiment results for the CIPE$_s$ scheme are not included, as its execution time is negligible compared to the DT-PKC cryptosystem. For example, the CIPE$_s$ scheme takes less than 1 s to retrieve IR from 1 million POIs.



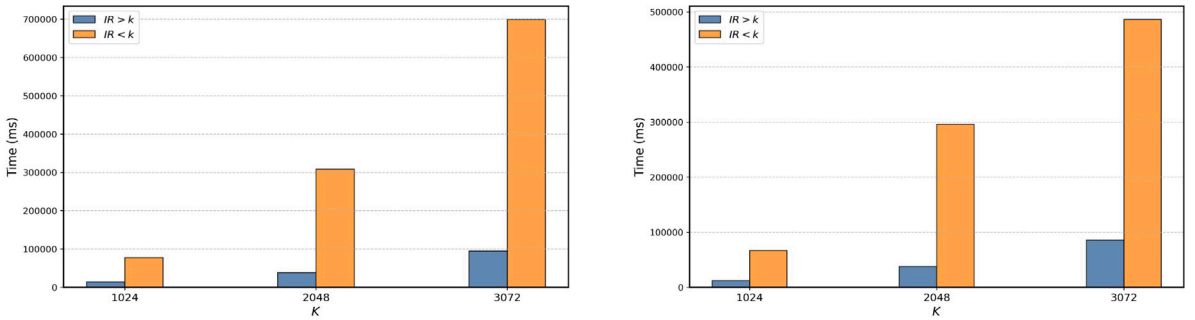
(a) n = 1024, k=10 and K ranges from 1024 to 3072 (California) (b) n = 1024, k=10 and K ranges from 1024 to 3072 (San Francisco)

Fig. 8. Comparison of search time between MSVkNN and DESMkNN on two datasets (K = 1024 to 3072).



(a) K = 1024, k=10 and n ranges from 1024 to 2024 (California) (b) K = 1024, k=10 and n ranges from 1024 to 2024 (San Francisco)

Fig. 9. Comparison of search time between MSVkNN and DESMkNN on two datasets (n = 1024 to 2024).



(a) n=1024, k=10 and K ranges from 1024 to 3072 (California) (b) n=1024, k=10 and K ranges from 1024 to 3072 (San Francisco)

Fig. 10. The search time of DESMkNN on two datasets (K = 1024 to 3072).

- *DO*: The execution time in data preprocessing are shown in Fig. 6. The computational cost includes two components: the cost of encrypting VD and the cost of generating SIG. Experiment results show that MSVkNN and SVkNN require additional operations such as grid partition, grid padding, and grid encryption, and thus perform worse in this stage.

- *Dual-Cloud Servers*: As shown in Section 7, the execution time in search stage is influenced by parameters n, k, K. Experiments are conducted under different parameter settings to demonstrate the effectiveness of DESMkNN. We can observe that the search time of DESMkNN is significantly shorter than MSVkNN, as shown in Figs. 7–9, primarily because MSVkNN incurs a high computational cost when executing the critical SGC protocol. Please note that in Fig. 7, both datasets (California Road Network and Points of Interest, San Francisco Road Network) are real-world datasets, where realistic POI distributions result in consistent performance gaps between DESMkNN and MSVkNN. Moreover, real-world datasets often exhibit a high density of POIs. Due to the grid partitioning mechanism, MSVkNN tends to be inefficient when handling real-world datasets. For example, in the California road network dataset, when setting the fine-grained grid parameter m in MSVkNN to 32 (which is the optimal parameter for MSVkNN), the number of POIs contained within each grid reaches as high as 108. To utilize data packing techniques, the parameter K needs to be adjusted to no less than 4096, which results in extremely high computational costs. However, in DESMkNN, well-designed data structures are employed to regulate the number of POIs

per partition, which keeps K within a reasonable range and prevents excessive computational overhead. As shown in Fig. 10, when IR is smaller than the query parameter k, the query time is significantly higher compared to when IR exceeds k, since CS need to perform more calculations related to homomorphic encryption. For a given scheme, larger values of k and n increase query time by expanding the search space and raising computational demands. Likewise, a larger K leads to longer plaintexts for encryption, which adds overhead from cryptographic operations.

In general, it can be concluded that DESMkNN not only meets the security requirements mentioned in Section 4 but also achieves higher efficiency than scheme MSVkNN in all stages of POI queries, with an improvement of up to 45.5%.

9. Conclusion

This paper proposes efficient and secure multi-user kNN queries with dynamic POIs updating, which preserves the privacy of data, queries, results, access patterns and ensures the results are correct and complete in a multi-user environment. Firstly, DESMkNN proposes a two-stage search framework to accelerate query speed. Secondly, DESMkNN designs a series of novel secure protocols and a compact verification strategy to facilitate the operation over the two-stage search framework. Finally, computational complexity, security analysis and experimental evaluation demonstrate that DESMkNN improves query efficiency by up tp 45.5% compared to MSVkNN. In future research, we plan to study kNN queries for multi-type POIs to address the limitation of single-type POI scenarios, where query results are too homogeneous. Moreover, we will focus more on exploring the balance between security and efficiency.

CRediT authorship contribution statement

Yining Jia: Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Yali Liu:** Writing – review & editing, Resources. **Congai Zeng:** Writing – review & editing. **Xujie Ding:** Writing – review & editing. **Jianting Ning:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank the editor and the reviewers for their comments and suggestions. This work was supported by the National Natural Science Foundation of China under Grant No. 61702237, No. 62425205, and No. 12441101, the Opening Foundation of State Key Laboratory for Novel Software Technology, Nanjing University under Grant No. KFKT2025B54, the Science and Technology Planning Foundation of Xuzhou City under Grant No. KC22052, the Opening Foundation of Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology under Grant GCIS202114, the Postgraduate Research & Practice Innovation Program of Jiangsu Normal University under Grant 2024XKT2579, and the University-Industry Collaborative Education Program of China under Grant No. 202101374001. All authors have read and approved the final version of the manuscript.

Data availability

Data will be made available on request.

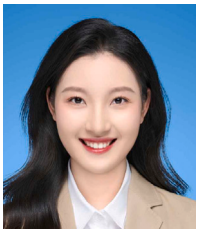
References

[1] R. Li, A. Liu, A. Wang, Fast and scalable range query processing with strong privacy protection for cloud computing, IEEE/ACM Trans. Netw. 24 (4) (2015) 2305–2318.
[2] G. Xiao, F. Wu, X. Zhou, K. Li, Probabilistic top-k range query processing for uncertain databases, J. Intell. Fuzzy Syst. 31 (2) (2016) 1109–1120.
[3] K. Xue, S. Li, J. Hong, Y. Xue, N. Yu, P. Hong, Two-cloud secure database for numeric-related SQL range queries with privacy preserving, IEEE Trans. Inf. Forensic Secur. 12 (7) (2017) 1596–1608.
[4] Y. Miao, Y. Yang, X. Li, K.-K.R. Choo, X. Meng, R.H. Deng, Comprehensive survey on privacy-preserving spatial data query in transportation systems, IEEE Trans. Intell. Transp. Syst. 24 (12) (2023) 13603–13616.
[5] Y. Zhang, B. Wang, Z. Zhao, Verifiable and privacy-preserving k-NN query scheme with multiple keys, IEEE Trans. Big Data 11 (3) (2024) 1434–1446.
[6] Q. Liu, Y. Peng, J. Wu, T. Wang, G. Wang, Secure multi-keyword fuzzy searches with enhanced service quality in cloud computing, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 2046–2062.
[7] Q. Liu, Y. Peng, Q. Xu, H. Jiang, J. Wu, T. Wang, T. Peng, G. Wang, S. Zhang, MARSMars: Enabling verifiable range-aggregate queries in multi-source environments, IEEE Trans. Dependable Secur. Comput. 21 (4) (2024) 1994–2011.
[8] N. Cui, X. Yang, B. Wang, J. Li, G. Wang, SVkNN: Efficient secure and verifiable k-nearest neighbor query on the cloud platform, in: Proc. of ICDE, 2020, pp. 253–264.
[9] N. Cui, K. Qian, T. Cai, J. Li, X. Yang, J. Cui, H. Zhong, Towards multi-user, secure, and verifiable k NN query in cloud database, IEEE Trans. Knowl. Data Eng. 35 (9) (2023) 9333–9349.
[10] H. Xie, Y. Guo, X. Jia, A privacy-preserving online ride-hailing system without involving a third trusted server, IEEE Trans. Inf. Forensics Secur. 16 (2021) 3068–3081.
[11] W. Wong, D. Cheung, B. Kao, N. Mamoulis, Secure kNN computation on encrypted databases, in: Proc. of SIGMOD, 2009, pp. 139–152.
[12] Y. Zhu, R. Xu, T. Takagi, Secure k-NN computation on encrypted cloud data without sharing key with query users, in: Proc. of IWSEC, 2013, pp. 55–60.
[13] B. Yao, F. Li, X. Xiao, Secure nearest neighbor revisited, in: Proc. of ICDE, 2013, pp. 733–744.
[14] Q. Liu, Z. Hao, Y. Peng, H. Jiang, J. Wu, T. Peng, G. Wang, S. Zhang, SecVKQ: Secure and verifiable kNN queries in sensor–cloud systems, J. Syst. Archit. 120 (2021) 102300.
[15] Y. Elmehdwi, B.K. Samanthula, W. Jiang, Secure k-nearest neighbor query over encrypted data in outsourced environments, in: Proc. of ICDE, 2014, pp. 664–675.
[16] S. Choi, G. Ghinita, H.-S. Lim, E. Bertino, Secure kNN query processing in untrusted cloud environments, IEEE Trans. Knowl. Data Eng. 26 (11) (2014) 2818–2831.
[17] K. Cheng, L. Wang, Y. Shen, H. Wang, Y. Wang, X. Jiang, H. Zhong, Secure k k-NN query on encrypted cloud data with multiple keys, IEEE Trans. Big Data 7 (4) (2021) 689–702.
[18] A. Boldyreva, N. Chenette, Y. Lee, A. O'neill, Order-preserving symmetric encryption, in: Proc. of EUROCRYPT, 2009, pp. 224–241.
[19] X. Liu, R.H. Deng, K.-K.R. Choo, J. Weng, An efficient privacy-preserving outsourced calculation toolkit with multiple keys, IEEE Trans. Inf. Forensics Secur. 11 (11) (2016) 2401–2414.
[20] K. Cheng, Y. Shen, Y. Wang, L. Wang, J. Ma, X. Jiang, C. Su, Strongly secure and efficient range queries in cloud databases under multiple keys, in: Proc. of INFOCOM, 2019, pp. 2494–2502.
[21] S.K. Nayak, S. Tripathy, SEMKC: Secure and efficient computation over outsourced data encrypted under multiple keys, IEEE Trans. Emerg. Top. Comput. 9 (1) (2018) 414–428.
[22] A. Okabe, B. Boots, K. Sugihara, S. Chiu, Spatial tessellations: Concepts and applications of voronoi diagrams, College Math. J. (2001).
[23] Y. Manolopoulos, A. Nanopoulos, A.N. Papadopoulos, Y. Theodoridis, R-Trees: Theory and Applications: Theory and Applications, Springer Science & Business Media, 2006.
[24] N. Cui, D. Wang, H. Zhu, J. Li, J. Xu, X. Yang, Enabling verifiable and secure range query in multi-user setting under cloud environments, IEEE Trans. Knowl. Data Eng. 36 (12) (2024) 8148–8163.
[25] Q. Liu, S. Wu, S. Pei, J. Wu, T. Peng, G. Wang, Secure and efficient multi-attribute range queries based on comparable inner product encoding, in: Proc. of CNS, 2018, pp. 1–9.
[26] Y. Zhang, B. Wang, Z. Zhao, Secure k-NN query with multiple keys based on random projection forests, IEEE Internet Things J. 11 (9) (2023) 15205–15218.
[27] S. Wu, Q. Li, G. Li, D. Yuan, X. Yuan, C. Wang, ServeDB: Secure, verifiable, and efficient range queries on outsourced database, in: Proc. of ICDE, 2019, pp. 626–637.
[28] H.-I. Kim, H.-J. Kim, J.-W. Chang, A secure kNN query processing algorithm using homomorphic encryption on outsourced database, Data Knowl. Eng. 123 (2019) 101602.
[29] A. Liu, K. Zhengy, L. Liz, G. Liu, L. Zhao, X. Zhou, Efficient secure similarity computation on encrypted trajectory data, in: Proc. of ICDE, 2015, pp. 66–77.

[30] P. Williams, R. Sion, B. Carbunar, Building castles out of mud: practical access pattern privacy and correctness on untrusted storage, in: Proc. of CCS, 2008, pp. 139–148.
[31] M.S. Islam, M. Kuzu, M. Kantarcioglu, Access pattern disclosure on searchable encryption: ramification, attack and mitigation, in: Proc. of NDSS, vol. 20, 2012, p. 12.
[32] A. Bowyer, Computing dirichlet tessellations, Comput. J. 24 (2) (1981) 162–166.
[33] D.F. Watson, Computing the n-dimensional delaunay tessellation with application to voronoi polytopes, Comput. J. 24 (2) (1981) 167–172.
[34] J. Liu, J. Yang, L. Xiong, J. Pei, Secure skyline queries on cloud platform, in: Proc. of ICDE, 2017, pp. 633–644.
[35] A.C.-C. Yao, How to generate and exchange secrets, in: Proc. of Sfcs, 1986, pp. 162–167.

Yining Jia received his B.Sc. in Computer Science and Technology in 2023 from Nanjing Forestry University, China. Currently, he is pursuing the M.Sc. degree in the School of Artificial Intelligence and Computer Science at Jiangsu Normal University, China. His research interests include data privacy, query processing, information security.

Yali Liu received her Ph.D. in 2014 from Nanjing University of Aeronautics and Astronautics, China. She is a senior member of China Computer Federation (CCF). She has been a Research Scientist at Nanyang Technological University, Singapore. She is currently a Professor in the School of Artificial Intelligence and Computer Science at Jiangsu Normal University, China. Her research interests include information security, authentication and privacy-preserving technology, blockchain security and privacy, vehicular ad hoc networks, cryptographic algorithms and protocols and their applications in Internet of things and mobile communication.



Congai Zeng received her M.Sc. in Electronic Information in 2024 from Jiangsu Normal University, China. Currently, she is pursuing the Ph.D. degree in the Faculty of Information Technology at Beijing University of Technology, China. Her research interests include Internet of Vehicles security and privacy.

Xujie Ding received his B.Sc. in Software Engineering in 2023 from Jiangsu Normal University, China. Currently, he is pursuing the M.Sc. degree in the School of Artificial Intelligence and Computer Science at Jiangsu Normal University, China. His research interests include privacy preservation and secure data sharing technology in smart healthcare.

Jianting Ning received his Ph.D. in 2016 from Shanghai Jiao Tong University, China. He has been a Research Scientist at the School of Computing and Information Systems, Singapore Management University, and a Research Fellow at the National University of Singapore. His research interests include applied cryptography and information security. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, China, and with Faculty of Data Science, City University of Macau, China. He has published papers in major conferences/journals, such as ACM CCS, NDSS, ASIACRYPT, ESORICS, ACSAC, IEEE Transactions on Information Forensics and Security, and IEEE Transactions on Dependable and Secure Computing.