

Article

Secure and Efficient Spatial Range Query Scheme over Multi-Type Encrypted Vector Data in the Cloud

Qi Yong ¹, Zhaoting Ma ^{1,*}, Dong Liu ¹, Xu Wang ¹, Jike Hui ² and Hao Li ¹

¹ Chinese Academy of Surveying and Mapping, Beijing 100036, China; a200102202021@163.com (Q.Y.); liudong@casm.ac.cn (D.L.); wx15670525424@163.com (X.W.); leonlee394@gmail.com (H.L.)

² Hebei GEO University, Shijiazhuang 052161, China; 17603275726@163.com

* Correspondence: mazht@casm.ac.cn

Abstract

Secure and efficient spatial range query over encrypted outsourced vector data remains a challenging issue, particularly for polyline and polygon data. Existing schemes primarily focus on point data and have not been properly extended to more complex spatial objects. Therefore, we propose a secure and efficient spatial range query scheme that supports multi-type encrypted vector data. The data owner first outsources encrypted vector data and the encrypted R-tree index to a cloud server, and then the cloud server retrieves matching data identifiers using an encrypted query rectangle and encrypted index. After that, the cloud server accesses the spatial database to obtain the corresponding encrypted vector data. Finally, the query user decrypts the encrypted vector data and filters with a query rectangle to get actual query results. Experiments show that our scheme achieves query efficiency comparable to that of query based on a plaintext index, while providing enhanced query security at the cost of only 3.97% performance overhead. We also compared our point data query with existing scheme. And the results indicate that our scheme effectively reduces false positives and improves query efficiency by enabling intersection tests between the query rectangle and the minimum bounding rectangles of spatial objects in the encrypted domain. Overall, our scheme can support spatial range query over multi-type encrypted vector data, achieving a balance between security and efficiency.

Keywords: data outsourcing; range query; encrypted index; encrypted vector data; data security

1. Introduction

With the rapid growth of data volumes, data owners tend to outsource their data to cloud servers to reduce local storage and computing costs [1]. As an honest-but-curious third party, cloud servers may analyze the data when executing query requests, thereby posing a risk of data leakage [2]. Therefore, original data is often encrypted using symmetric encryption algorithms before being outsourced. However, symmetric encryption disrupts the structure and order of the data, making it difficult to directly execute query operations that rely on this information [3]. Although encrypted data files can be sent to query users for local decryption and query, this increases the computational burden on users and transmits a large amount of redundant data, thereby reducing query efficiency and wasting network resources. Therefore, it is of great significance to develop a spatial query scheme that balances security, efficiency, and user usability.

Academic Editor: Rui Araújo

Received: 8 June 2026

Revised: 28 June 2026

Accepted: 29 June 2026

Published: 2 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Query service is a basic requirement in data outsourcing, and its form exhibits diversity depending on the various query needs [4–7]. Vector data is widely used in geographic information systems and is a common type of outsourced data [8]. As a basic operation in vector data analysis, spatial range query plays an important role in domains including urban planning, transportation and logistics, natural resource management, and location-based services, as well as other related areas [9]. For example, in urban planning and management, administrative departments can retrieve geographic features within a specific area through spatial range query and analyze their spatial distribution, which helps support resource allocation decisions. However, outsourced data is generally encrypted, so cloud servers cannot construct a meaningful spatial index on encrypted data for range query. In addition, cloud servers are also unable to decrypt outsourced data to perform a query on the original data. Therefore, how to support spatial range query over encrypted vector data becomes a key issue.

Regarding the spatial range query on encrypted vector data, existing research has primarily adopted a strategy combining cryptography and index [10–17]. On the one hand, symmetric encryption algorithms (such as AES and SM4) [18,19] are utilized to protect the original vector data. On the other hand, an encrypted index is constructed to enable fast query, allowing the cloud server to perform a spatial range query without decrypting the vector data. This strategy balances data security and query efficiency. However, existing schemes are primarily designed for point data and offer limited support for polyline and polygon data, making it difficult to meet the query requirements for multiple types of vector data. Point data cannot convey the shape and spatial extent of objects. If points are used to represent spatial objects such as buildings or roads, query accuracy can be significantly affected. Hence, the need to support query over polyline and polygon data cannot be ignored.

To address the above issue, we propose a secure and efficient spatial range query scheme that supports multi-type encrypted vector data. Specifically, we employ Order-Preserving Encryption (OPE) and Sort-Tile-Recursive (STR) algorithm to construct an encrypted R-tree index, and utilize the symmetric encryption algorithm SM4 to process the original vector data. Subsequently, the encrypted index and data are outsourced to the cloud server for store and query, with the index stored in the cloud server's hard disk and the data stored in a spatial database deployed on the cloud server. Our contributions are as follows:

- We propose a secure and efficient spatial range query scheme for multi-type encrypted vector data, which supports query processing for point, polyline and polygon data.
- Only a single round of communication is required between the trusted user and the cloud server, avoiding frequent data transmission.
- We conduct query experiments on eight datasets consisting of point, polyline, and polygon data, and compare the query efficiency of our scheme with those of the comparison schemes.
- The experimental results show that, compared to plaintext index query, the maximum efficiency loss of our scheme is only 3.97%, achieving a balance between query efficiency and security.

The rest of this paper is organized as follows. In Section 2, we provide an overview of related work on spatial range query for encrypted vector data. In Section 3, we introduce some techniques used in our scheme. In Section 4, we describe the system model, problem definition, and design goals. Then, in Section 5, we present the implementation details of our scheme. Next, we show the security analysis in Section 6 and present the performance analysis in Section 7. Finally, in Section 8, we draw a conclusion of our work.

2. Related Work

Existing research on spatial range query for encrypted vector data often employs cryptographic primitives such as Order-Preserving Encryption and Homomorphic Encryption to ensure that cloud servers can perform query without decrypting the data [20]. To improve query efficiency and reduce computational overhead, existing schemes often combine encryption technology with spatial query optimization technology. Bloom filter [21], space filling curve [22], and learned index [23] are common query optimization methods.

A Bloom filter can quickly check whether an element belongs to a set and is often used to filter out irrelevant data. Wang et al. [11] proposed the Geometric Range Searchable Encryption (GRSE) scheme using a Bloom filter and Shen-Shi-Waters encryption (SSW), which enabled the retrieval of point data within an arbitrary geometric range. However, GRSE does not consider the redundant bits that do not map information in the Bloom filter, resulting in huge computational and storage overhead. To address this problem, Miao et al. [12] designed the Privacy-Preserving Spatial Range Query (PSRQ) scheme, which constructed a Circular Shift and Coalesce Bloom Filter (CSC-BF) and used Symmetric-Key Hidden Vector Encryption (SHVE) to encrypt the valid bits mapped to the CSC-BF, thereby reducing computational and storage costs. Miao et al. [24] also designed the Efficient and Secure Spatial Range Query (ESSRQ) scheme based on the Geohash algorithm, CSC-BF, and Symmetric Homomorphic Encryption (SHE), making the query complexity independent of dataset size and significantly improving the query efficiency for large-scale point data. However, the above schemes based on Bloom filter [11,12,24] still face challenges when handling polyline and polygon data. On the one hand, if polyline or polygon is simplified into a single point and then mapped into the Bloom filter, the loss of spatial information results in a large number of false positives (irrelevant data). On the other hand, if all the codes covered by polyline or polygon are mapped into the Bloom filter, the storage cost will increase greatly, and the higher probability of hash collisions will reduce the query accuracy.

Space-filling curve (SFC) can map multidimensional data into one-dimensional space, hence transforming complex two-dimensional geometric query into one-dimensional numerical interval query. Kim et al. [25] proposed the Hilbert-curve-based Cryptographic Transformation (HCT) scheme, which enabled efficient range query for encrypted point data through Hilbert aggregation index. However, the HCT scheme requires multiple rounds of communication, resulting in increased communication overhead. Subsequently, Kamel et al. [10] proposed the Dynamic Index for Spatial data on the Cloud (DISC) scheme, which utilized a Hilbert R-tree index protected by hybrid encryption to enable point data query in a single round of communication. However, SFC struggles to preserve the topological integrity of spatial objects when processing polyline and polygon. Polyline and polygon objects are typically mapped to encoding intervals that span multiple Hilbert grids, and this discrete representation leads to index bloat [26]. Furthermore, the mismatch between encoding intervals and actual geometric shapes introduces a vast amount of irrelevant candidate data, thereby reducing query efficiency. In the data outsourcing environment, this will increase communication and client-side filtering burden, which in turn limits the practicality of the scheme on large-scale vector datasets.

Traditional tree-based index query schemes (such as R-tree and Quadtree) often cause redundant computational costs because each query needs to access the encrypted data stored by all nodes on the path from the tree root to leaf node [27]. To achieve better query performance, researchers utilized learned index to quickly predict the storage locations of data. Wang et al. [28] designed the learned Z-order Model (ZM) index, and proved that in most scenarios, the learned index performed more efficiently than R-tree. However, the ZM index does not consider the issue of privacy protection. Therefore, Li et al. [27]

proposed an efficient conjunctive geometric range query scheme with a learned index, which achieved differential privacy protection by adding noise to the objective function. Wang et al. [29] proposed the SLS-index-based Range Queries (SLRQ) scheme, which utilized the Paillier cryptosystem and secure learned spatial index (SLS-index) to achieve efficient data bucket filtering while ensuring strong security. However, most existing learned index schemes for encrypted data rely on spatial-filling curve to map multidimensional data to one dimension, and are essentially still designed for point data [28]. The above schemes [27–29] all utilize Z-order curve during the construction of learned indexes. The core function of spatial-filling curve is to map spatial data to a unique one-dimensional value. However, polyline and polygon data often intersect with multiple grids, making it difficult to accurately represent them using a unique code [30].

To meet the spatial range query requirements for multi-type encrypted vector data, and considering query efficiency and user burden, we propose a secure index scheme based on R-tree. Unlike Bloom filter and SFC, our scheme utilizes the minimum bounding rectangle (MBR) to uniformly represent heterogeneous spatial objects. This approach avoids fragmented segmentation and excessive simplification of continuous spatial objects, ensuring the integrity and uniqueness of spatial object representations. Based on this unified representation, all spatial objects—regardless of their original type—are indexed using an encrypted R-tree constructed over MBRs. During query processing, the encrypted query rectangle is compared against encrypted MBRs to determine candidate objects in a uniform manner.

3. Preliminaries

This section will introduce the main techniques supporting our scheme, including Order-Preserving Encryption, Sort-Tile-Recursive algorithm, and R-tree index.

3.1. Order-Preserving Encryption

The core characteristic of Order-Preserving Encryption (OPE) is the fact that the ciphertext maintains the same numerical order as the original data, thereby enabling efficient comparison calculation and range query on encrypted data [31]. (If $a_1 > a_2$, then $E_{OPE}(a_1) > E_{OPE}(a_2)$).

Our scheme encrypts R-tree and query rectangle through the provably secure Order-Preserving Encryption proposed by Boldyreva et al. [31]. Boldyreva et al. [31] were the first to provide a complete security analysis on OPE. During query processing, the cloud server can determine whether the encrypted index node intersects with the encrypted query rectangle without data decryption.

3.2. R-Tree Index

R-tree index is a balanced tree structure used for retrieving spatial data [32]. R-tree achieves efficient data retrieval by organizing the MBRs of spatial objects into a hierarchical tree structure. While querying, the cloud server starts from the root node and checks whether the MBR of each node intersects with the query rectangle, and recursively visits child nodes that meet the condition.

3.3. Sort-Tile-Recursive Algorithm

The Sort-Tile-Recursive (STR) algorithm is a spatial index construction algorithm that can be used to efficiently construct R-tree [33]. This algorithm can ensure that spatially adjacent objects are placed in the same node as much as possible through sorting and tiling, thereby reducing node overlap and improving query efficiency.

The first step of index construction is to obtain the center point coordinates (X_c, Y_c) of the MBR of each spatial object. Next, the spatial data is sorted by X_c and divided into

multiple tiles. Within each tile, the data is sorted by Y_c . After that, leaf nodes are generated according to node capacity. Finally, the above operations are recursively performed on the leaf nodes to construct upper-level nodes until the root node is formed.

4. Problem Formulation

In this section, we present the system model, problem definition, and design goals.

4.1. System Model

The system model of the proposed scheme, as shown in Figure 1, consists of three entities: the data owner (DO), trusted user (TU), and cloud server (CS).

- **Data Owner:** Encrypts vector data and index using SM4 and OPE respectively in the trusted environment, and then outsources them to the CS. The DO is responsible for the secure storage of the vector data encryption key K_{SM4} , initial vector and the index encryption key K_{OPE} . These parameters are distributed to the TU through a secure channel protected by transport security protocol such as TLS.
- **Trusted User:** Encrypts the query rectangle using the K_{OPE} and submits encrypted query request to the CS. After obtaining the query results returned by CS, TU decrypts the query results using K_{SM4} and filters out false positives (irrelevant data) in the plaintext domain.
- **Cloud Server:** Stores the encrypted data and index outsourced by DO, and provides encrypted data range query services to TU.

In the system, all vector data, index, query rectangles and query results are transmitted in encrypted form, ensuring data confidentiality.

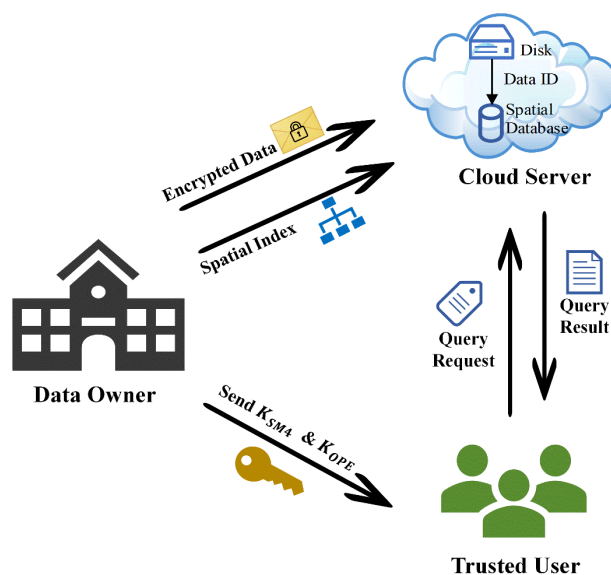


Figure 1. System model for spatial data outsourcing.

4.2. Problem Definition

Given a vector dataset $DB = \{O_1, O_2, \dots, O_i, \dots, O_s\}$, where O_i represents any spatial object, and s denotes the total number of spatial objects in the dataset, our scheme comprises five algorithms:

$$\Pi = (Index_{enc}, SpatialData_{enc}, QueryRange_{enc}, Query_{CS}, DecFilter_{TU}).$$

The algorithm is defined as follows:

- $I_{enc} \leftarrow Index_{enc}(DB, K_{OPE})$: DO runs the algorithm to generate an encrypted spatial index for each vector dataset.
- $DB_{enc} \leftarrow SpatialData_{enc}(DB, K_{SM4})$: DO encrypts the original spatial objects to provide strong data privacy protection.
- $QR_{enc} \leftarrow QueryRange_{enc}(QR, K_{OPE})$: TU performs OPE on the query rectangle to ensure that range query can be executed in the ciphertext domain.
- $R_{CS} \leftarrow Query_{CS}(I_{enc}, QR_{enc}, DB_{enc})$: CS processes the query request and returns the encrypted query result R_{CS} to TU.
- $R_{TU} \leftarrow DecFilter_{TU}(R_{CS}, QR, K_{SM4})$: TU decrypts R_{CS} and filters out false positives to obtain actual query results.

The notations used in the paper and their descriptions are shown in Table 1.

Table 1. Notations and descriptions.

Notations	Descriptions
K_{SM4}	Vector data encryption key
K_{OPE}	Index and query rectangle encryption key
DB_{enc}	Encrypted vector dataset
I_{enc}	Encrypted spatial index
QR_{enc}	Encrypted query rectangle
R_{CS}	Query results provided by CS
R_{TU}	Actual query results
n_i	Index tree node

4.3. Design Goals

The design goals of our scheme are to achieve secure spatial range query for multi-type encrypted vector data, specifically satisfying the following requirements:

- **Query Privacy:** Vector data is encrypted using a symmetric encryption algorithm before being outsourced, so CS cannot obtain the original data and its distribution pattern. TU also do not know extra data information except for query results. In addition, CS cannot access the actual query rectangle because it has been encrypted by TU.
- **Query Efficiency:** Index retrieval only requires low-complexity comparison calculations. Furthermore, the query process can be completed with single round of communication.
- **Query Accuracy:** The query results returned by CS include all spatial objects that intersect with the query rectangle. Although R_{CS} may contain false positives, no spatial objects that are actually within the query rectangle will be omitted.

5. Our Proposed Scheme

This section provides implementation details of our scheme, comprising the following three parts: index construction and vector data process, cloud server query, and trusted user decryption and filtering. Figure 2 shows the complete flow chart of the scheme proposed in this paper.

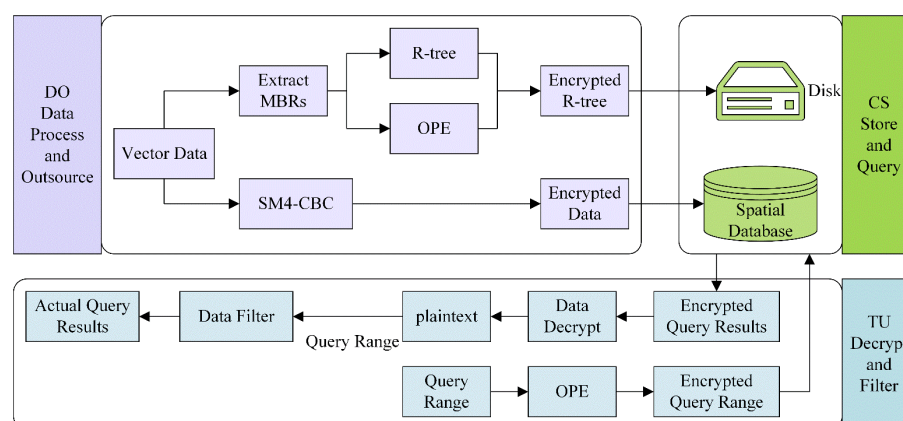


Figure 2. Flow chart of proposed scheme.

5.1. Index Construction and Vector Data Process

Our scheme is designed for point, polyline and polygon vector data. Typically, each polyline or polygon record is composed of a large number of points. In order to minimize the index size and maintain data security, each data entry only stores the unique identifier and encrypted MBR of spatial object. The use of MBRs to replace original vector data offers two mainly advantages. First, as abstract representation of original spatial objects, MBRs provide only the boundary of spatial objects, thereby reducing the risk of precise location leakage. Second, compared with original polyline or polygon, MBRs require significantly less storage space, which effectively prevents excessive index bloat and improves query efficiency. Algorithm 1 gives the pseudocode for encrypted index construction.

Algorithm 1: Encrypted index construction (by DO)

```

1: Input:
2:   Spatial dataset  $DB = \{O_1, O_2, \dots, O_i, \dots, O_s\}$ 
3:   Encryption key  $K_{OPE}$ 
4: for each  $O_i$  in  $DB$  do
5:   Extract the MBR of  $O_i \in DB$ 
6:   Store MBR in set  $\mathcal{M}$ 
7: end for
8: Construct a plaintext R-tree  $I$  using STR on  $\mathcal{M}$ 
9: for each MBR in  $I$  do
10:  Encrypt MBR using  $K_{OPE}$ 
11: end for
12: Return encrypted R-tree

```

We employ the STR algorithm to construct R-tree index, thereby reducing overlap between nodes. For MBRs in the R-tree, we encrypt it using OPE to prevent the disclosure of the original data distribution. As OPE is applied per dimension, all the X and Y coordinates are encrypted independently. Specifically, each coordinate component is treated as an individual value and encrypted separately. This reveals only one-dimensional order information and distribution characteristics along the X and Y axes, but does not expose the exact coordinate values or full two-dimensional geometric structure. Figure 3 illustrates the construction of the encrypted R-tree index. To balance the depth of the index tree with query efficiency, we first set the depth of R-tree to 3, thereby determining that the node capacity M is also 3 in Figure 3. Using the formula $\lceil \sqrt{s/M} \rceil$, we calculate that the data space can be divided into two tiles [33]. Considering the spatial distribution of the vector data, we divide the data in Figure 3 into two tiles, namely N1 and N2. Within

each tile, considering the numerical magnitude of Y_c and the node capacity, we obtain four leaf nodes: N3, N4, N5, and N6.

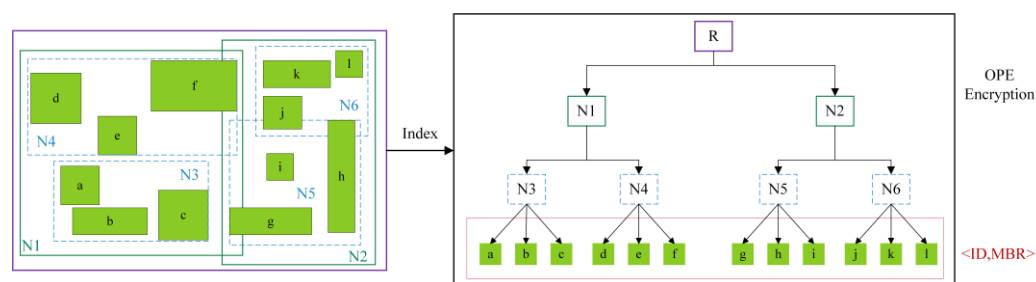


Figure 3. The construction of the encrypted spatial index.

After the construction of the R-tree index, it is time to process the original vector data. Generally, vector data is represented precisely by multiple pairs of coordinates, which determine the data's geographic location and shape in the real world [34]. The disclosure of coordinates can pose significant privacy and security risks. Therefore, it is necessary to encrypt the coordinates before outsourcing the data to ensure confidentiality during transmission and storage. We use SM4-CBC to encrypt the coordinates and Figure 4 shows the CBC encryption mode. In CBC encryption mode, since a different initialization vector is used for each encryption operation and the ciphertext is influenced by the previous ciphertext block, the same plaintext results in different ciphertext after encryption [35]. Even if an attacker obtains multiple ciphertexts, they cannot easily decrypt them or infer the original data.

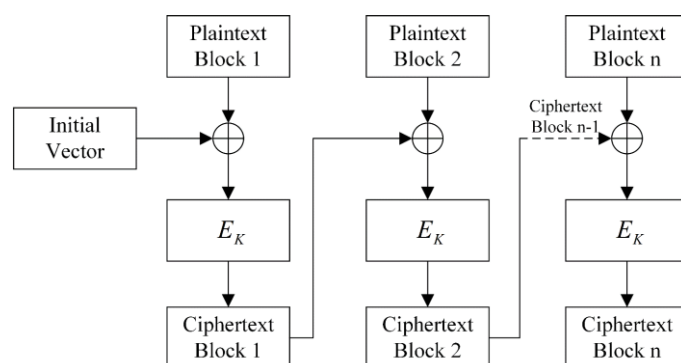


Figure 4. The CBC encryption mode.

After completing the above tasks, the encrypted vector data and the corresponding encrypted index need to be outsourced to CS. Since our scheme does not store the original vector data within the index, it is necessary to access the outsourced encrypted vector data to obtain the corresponding encrypted query results according to the identifiers returned by the index retrieval. However, common vector data formats (such as GeoJSON, KML, and GML) typically rely on sequential parsing and lack efficient random access mechanisms, thereby impairing query performance [36–38]. For Shapefile data, the original .shx file, which relies on fixed offsets, becomes invalid due to changes in data length caused by padding resulting from encryption [39]. To address this issue, a spatial database is deployed on the CS, and the encrypted vector data are imported into it. When performing a query, the unique identifier of each spatial object is used as a linkage key, enabling the cloud server to efficiently retrieve the corresponding encrypted vector data through database lookup operations.

5.2. Cloud Server Query

During the query process, the cloud server starts from the root node to determine whether QR_{enc} intersects with the MBR of the current node. If there is an intersection, the search recursively continues to the child nodes until the leaf nodes are reached and checks each data entry. If there is no intersection, the search along that path is terminated. Since the X and Y coordinates are encrypted separately, the intersection test is performed by independently comparing the coordinate values along the X and Y axes to identify the data identifiers that satisfy the query conditions. The specific judgment logic is as follows:

$$!(node_mbr.min_x > query_rect.max_x \parallel node_mbr.max_x < query_rect.min_x \parallel node_mbr.min_y > query_rect.max_y \parallel node_mbr.max_y < query_rect.min_y),$$

where $node_mbr$ denotes the encrypted MBR of an index node, and $query_rect$ denotes the encrypted query rectangle.

The CS first filters out matching spatial object identifiers through I_{enc} and QR_{enc} , and then interacts with the spatial database to extract the encrypted vector data corresponding to the identifiers. Since the data is encrypted before outsourcing, the query results are returned to TU in encrypted form. TU decrypts and filters the results to obtain the actual query results. Algorithm 2 provides the pseudocode for the cloud server query.

Algorithm 2: Spatial query process on encrypted data

```

1:  Input:
2:    Query range  $QR = [(x_{min}, y_{min}), (x_{max}, y_{max})]$ 
3:    Encryption key  $K_{OPE}$ 
4:     $QR_{enc} \leftarrow \text{Encrypt } QR \text{ using } K_{OPE}$  ▷ At TU
5:    Send  $QR_{enc}$  to CS
6:    for all  $n_i$  in  $I_{enc}$  do
7:      if  $n_i \cap QR_{enc} \neq \emptyset$  then ▷ At CS
8:        if  $n_i$  is leaf then
9:          Search each data entry and add matching data to  $R_{CS}$ 
10:       else if  $n_i$  is internal node then
11:         Search each child of  $n_i$ 
12:       end if
13:     end if
14:   end for
15:   Return  $R_{CS}$  to TU

```

5.3. Trusted User Decryption and Filtering

The MBR-based intersection test method may cause false positives. Figure 5 illustrates false positives for polyline and polygon data in detail. Figure 5a,b show cases of false positives in polyline data, while Figure 5c,d show cases of false positives in polygon data. It can be seen that even if the spatial objects are disjoint from the query rectangles, the MBRs still intersect with the query rectangles, hence resulting in false positives. To address false positives in query results, TU has to decrypt the query results returned by CS and filter out mismatched vector data. Algorithm 3 displays the pseudocode for query result R_{CS} decryption and filtering.

Algorithm 3: Decrypt-and-filter query result R_{CS} (by TU)

```

1:  Input:
2:    Query results  $R_{CS}$  returned by cloud server
3:    Encryption key  $K_{SM4}$ , initial vector, original query rectangle  $QR$ 
4:  for each record in  $R_{CS}$  do
5:    Decrypt encrypted record using  $K_{SM4}$  and initial vector

```

```

6: end for
7: for each decrypted record do
8:     Filter the decrypted record with  $QR$  and discard the mismatched record
9: end for
10: Return actual query results  $R_{TU}$ 

```

There are three spatial relationships between point data and the query rectangle: point inside the query rectangle, point outside the query rectangle, and point on the boundary of the query rectangle. In range query, these three relationships can be accurately determined through coordinates, thus avoiding false positives.

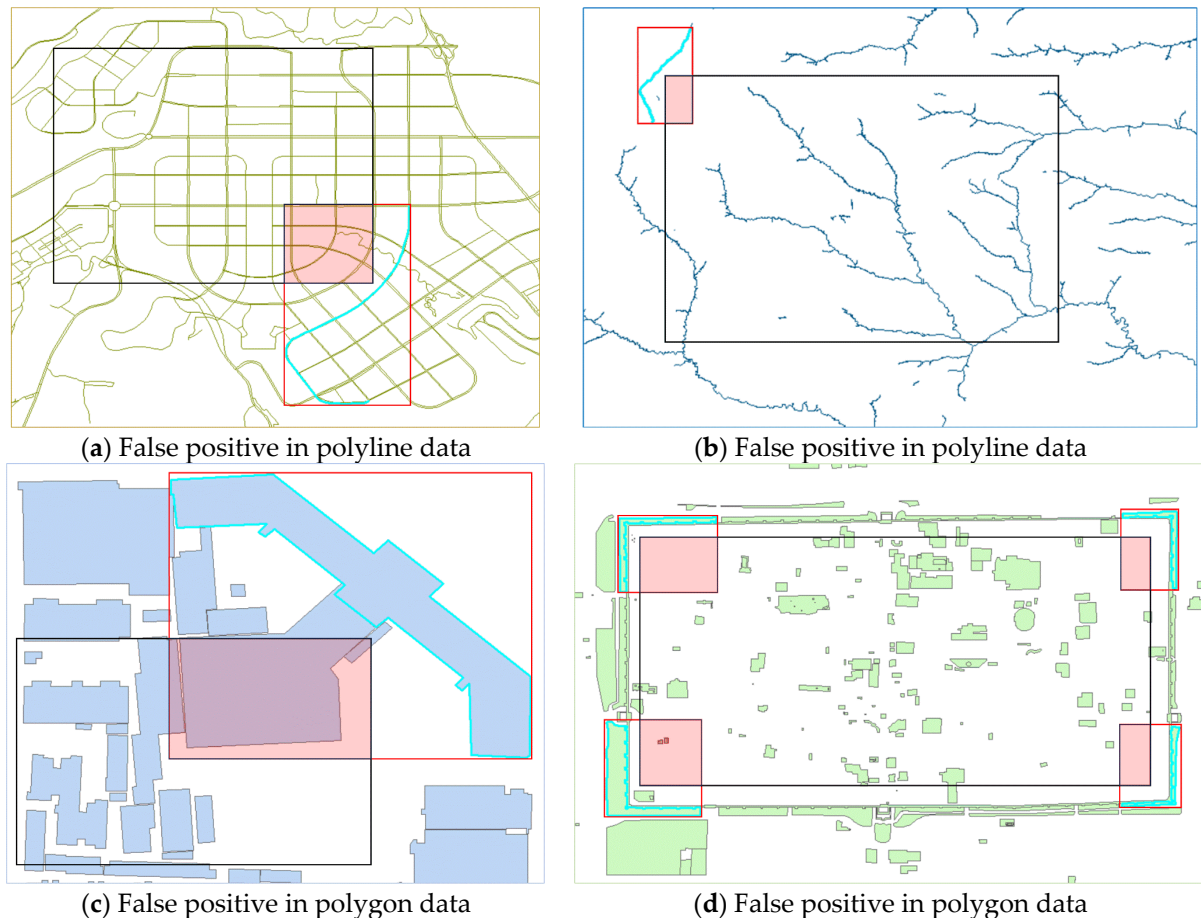


Figure 5. The cases of false positives. Note: The highlighted polylines or polygons are actual spatial objects. The red rectangles are the MBRs of spatial objects and the black rectangles are the query rectangles. The pink areas indicate the intersection between the MBRs and the query rectangles.

6. Security Analysis

6.1. Leakage Function

We first define the leakage function. The leakage caused by query Q on dataset DB is denoted as $\mathcal{L}(DB, Q)$, which specifically includes the following aspects:

- **Size Pattern:** CS can learn the total number of data records in the dataset.
- **Access Pattern:** CS knows the identifier of each encrypted data record returned by each spatial range query.
- **Search Pattern:** CS knows whether the same encrypted data record was retrieved via two different queries.

6.2. Data Security

Our scheme is suitable for multiple types of vector data. Specifically, the data security for polyline and polygon data is guaranteed under IND-CPA, while that for point data and encrypted R-tree index are established under POPF-CCA [40,41].

Before the formal data security analysis for polyline and polygon data, we first simulate a security game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$.

Definition 1. Let $\Pi = (Index_{enc}, SpatialData_{enc}, QueryRange_{enc}, Query_{CS}, DecFilter_{TU})$ be a spatial range query scheme for multiple types of encrypted vector data over security parameters λ . The security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is defined as:

Init: Given security parameter λ , the adversary $\mathcal{A}$ generates two datasets $DB_1 = \{O_{1,1}, O_{1,2}, \dots, O_{1,s}\}$ and $DB_2 = \{O_{2,1}, O_{2,2}, \dots, O_{2,s}\}$ with the same number of spatial data objects as $\mathcal{C}$.

Phase 1: $\mathcal{A}$ adaptively submits a number of requests, where each request is one of the two following types:

- **Ciphertext request:** On the j th ciphertext request, adversary $\mathcal{A}$ outputs a dataset $DB_j = \{O_{j,1}, O_{j,2}, \dots, O_{j,s}\}$, and challenger $\mathcal{C}$ returns encrypted index $I_{enc,j}$ through $Index_{enc}$.
- **Encrypted query range request:** On the j th query request, adversary $\mathcal{A}$ outputs a spatial query range QR_j , and challenger $\mathcal{C}$ responds with encrypted query range $QR_{enc,j}$ through $QueryRange_{enc}$.

Challenge: With DB_1 and DB_2 selected in **Init**, challenger $\mathcal{C}$ flips a coin $b \in \{0,1\}$ and returns $I_{enc,b}$ of DB_b to adversary $\mathcal{A}$.

Phase 2: Adversary $\mathcal{A}$ continues to adaptively submit a number of requests, which are still subjected to the same restrictions in **Phase 1**.

Guess: Adversary $\mathcal{A}$ gives a guess b' of b .

We say that our scheme is secure against IND-CPA on data security for polyline and polygon data if for any polynomial-time adversary in the above game, it has at most a negligible advantage.

$Adv_{Our\ scheme, \mathcal{A}}^{IND-CPA-Data}(1^\lambda) = \left| Pr[b' = b] - \frac{1}{2} \right| \leq negl(\lambda)$, where $negl(\lambda)$ is the negligible function in λ .

We now analyze the data security for polyline and polygon data. Since the original vector data in our solution is encrypted using SM4, data security relies on the security of a symmetric encryption algorithm.

Theorem 1. Polyline and polygon data encrypted with SM4 achieve IND-CPA security because SM4 is IND-CPA-secure.

Proof. To prove the IND-CPA data security of our scheme for polyline and polygon data, we simulate the security game defined in **Definition 1** with an adversary $\mathcal{A}$ from the ideal security game of SM4. For each object in DB_j , the MBRs are first extracted to construct an index, and then encrypted using SM4. In **Phase 1** and **Phase 2** of the security game, $\mathcal{A}$ chooses different $O_{j,i}$ each time and observes the corresponding ciphertext in $I_{enc,j}$. Since the key K_{SM4} is kept secret from $\mathcal{A}$ and the encryption follows the standard SM4-CBC, the probability of $\mathcal{A}$ distinguishing between two ciphertexts can be reduced to breaking the underlying SM4. Furthermore, although the MBRs are encrypted via OPE for index construction, the MBR only provides an approximate boundary and does not leak the precise vertex coordinates of polyline and polygon data. Namely, for any pair of messages and the corresponding ciphertext chosen by $\mathcal{A}$, $\mathcal{A}$ cannot determine which message has been encrypted. Then we have $Adv_{Our\ scheme, \mathcal{A}}^{IND-CPA-Data}(1^\lambda) = \left| Pr[b' = b] - \frac{1}{2} \right| \leq$

$\text{negl}(\lambda)$, which proves polyline and polygon data meet IND-CPA security in our scheme. $\square$

The OPE adopted in the paper is proposed by Boldyreva et al., and is based on the hypergeometric distribution, namely OPE^{HG} . OPE^{HG} consists of an encryption algorithm Enc^{HG} , and a decryption algorithm Dec^{HG} , and achieves order-preserving mapping from plaintexts to ciphertexts through recursive binary partitioning and hypergeometric sampling. The random bits required during the sampling process are generated by a pseudorandom function TapeGen, which is composed of VIL-PRF (Variable Input-Length Pseudorandom Function) and VOL-PRG (Variable Output-Length Pseudorandom Generator), both constructed based on AES.

The OPE^{HG} achieves POPF-CCA security. In the ideal setting, the stateful LazySample algorithm, supplied with truly random bits, generates a uniformly random order-preserving function. OPE^{HG} differs from LazySample only in that the true random source is replaced by TapeGen. Therefore, the behavioral difference between the two algorithms is solely determined by the distinguishability between the pseudorandom bits and truly random bits. Formally, the security bound is given by $\text{Adv}_{OPE^{HG}}^{\text{popf-cca}}(A) \leq \text{Adv}_{\text{TapeGen}}^{\text{popf-cca}}(B) + \lambda$, where adversary $\mathcal{A}$ is against OPE^{HG} and adversary $\mathcal{B}$ is against TapeGen. The parameter λ is a negligible constant independent of the sizes of the plaintext and ciphertext domains, arising from the floating-point precision loss incurred during hypergeometric sampling. If there exists an adversary $\mathcal{A}$ capable of distinguishing OPE^{HG} from an ideal random order-preserving function, then a corresponding adversary $\mathcal{B}$ can be constructed to break the pseudorandomness of TapeGen. By contraposition, if TapeGen satisfies the pseudorandomness property, then OPE^{HG} is secure under the POPF-CCA security notion.

The security of TapeGen can be further reduced to that of the underlying AES. Specifically, the pseudorandomness of both the VIL-PRF and the VOL-PRG relies on the PRF security of AES. If the output of AES is computationally indistinguishable from a uniformly random bit string, then the $\text{Adv}_{\text{TapeGen}}^{\text{popf-cca}}(B)$ is negligible. Consequently, after accounting for the negligible error λ , the $\text{Adv}_{OPE^{HG}}^{\text{popf-cca}}(A)$ is also negligible. Thereby OPE^{HG} achieves POPF-CCA security.

For point data, the MBRs degenerate to coordinates, and thus index entries store the coordinates encrypted using OPE. Since OPE is POPF-CCA-secure, point data and MBRs also achieve POPF-CCA security.

6.3. Query Security

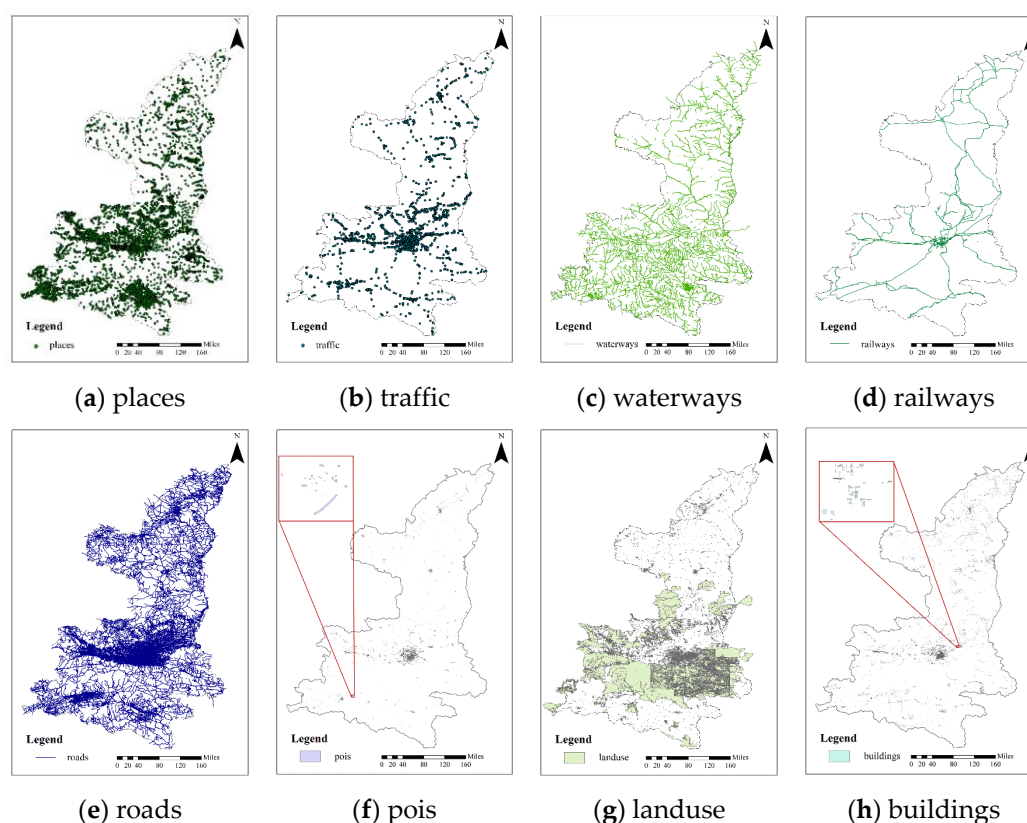
CS can infer which spatial objects are frequently accessed by observing the query results. However, CS can only obtain the data identifiers and encrypted vector data instead of original vector data corresponding to the identifiers. Since the query rectangle is also encrypted, CS cannot know the user's actual query area. In addition, only TU can obtain K_{SM4} , and attackers cannot decrypt the data without this key.

7. Performance Analysis

This section evaluates the performance of our scheme by displaying the query rectangle encryption process, query process efficiency, and point data query comparison. The experiment utilizes some vector datasets from Shaanxi Province (downloaded from the OpenStreetMap website), and the detailed information of each dataset is shown in Table 2. Figure 6 shows the spatial distribution of each dataset. All the experiments are conducted using C++ on a 64-bit Windows 10 operating system, and are completed on an Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz server. The database platform and version information is PostgreSQL 10.17.

Table 2. Detailed information of datasets.

Data Types	Datasets	File Size	Number of Records	Number of Vertices
point	<i>places</i>	214 KB	7789	7789
point	<i>traffic</i>	584 KB	21,339	21,339
polyline	<i>waterways</i>	6621 KB	7881	396,113
polyline	<i>railways</i>	2411 KB	15,749	99,156
polyline	<i>roads</i>	56,609 KB	184,166	2,978,370
polygon	<i>pois</i>	1767 KB	7744	85,919
polygon	<i>landuse</i>	49,807 KB	77,868	2,914,401
polygon	<i>buildings</i>	37,950 KB	173,857	1,819,990

**Figure 6.** The spatial distribution of each dataset.

7.1. Query Rectangle Encryption Process

To enhance query security, our scheme adopts an encrypted index. Therefore, the query rectangle must be encrypted using OPE before the query is performed. We encrypt the coordinates of the query rectangle 100 times and record the time cost for each encryption. Finally, we calculate the average process time for the query rectangle encryption is 170 μ s.

7.2. Query Process Efficiency

Regarding query process efficiency, our scheme is compared with a scheme that constructs a plaintext index for vector data and outsources it for query. The query process of the comparison scheme is consistent with that of our scheme, but the R-tree index and query rectangle do not need to encrypt. We perform spatial range query on each dataset, but due to space limitations, we only display the experimental results of *traffic*, *waterways* and *buildings* in detail, while the results for the remaining datasets are reported only for the 15% query size. Tables 3–5 show the query process information for both schemes un-

der the same dataset and query size in detail. Table 6 shows the summary of experimental results on remaining datasets. Each query runs 100 times and takes the average as the final query process time. Figures 7–9 show the query range and query results for *traffic*, *waterways* and *buildings* at 1% query size, which demonstrate the feasibility of our scheme and the accuracy of the query results. For three different types of datasets—point, polyline and polygon—the query process time of our scheme is comparable to that of the comparison scheme. Since both schemes use the same index construction algorithm, the index retrieval path and cloud server query results are the same. Therefore, the communication cost of the query results and the TU decryption and filtering cost are the same for both schemes. Taking the comparison scheme as the baseline, the maximum query efficiency loss of our scheme is 3.97% according to the formula $\left(1 - \frac{\text{Query Range Encrypt} + \text{CS Query} + \text{TU Encrypt \& Filter}}{\text{Comparison Scheme} + \text{TU Encrypt \& Filter}}\right) \times 100\%$. Considering query efficiency and security, this loss is acceptable.

Table 3. Query process information of *traffic*.

Query Size	Our Scheme		Comparison Scheme/ms	TU Decrypt & Filter/ms	Communication Cost/KB	Efficiency Loss/%
	Query Range Encrypt/ms	CS Query/ms				
1%	0.17	1.09	1.01	5.28	7.49	−3.97
5%	0.17	3.80	3.76	16.40	37.51	−1.04
10%	0.17	6.61	6.43	28.11	75.02	−1.01
15%	0.17	9.39	9.41	40.64	112.54	−0.30

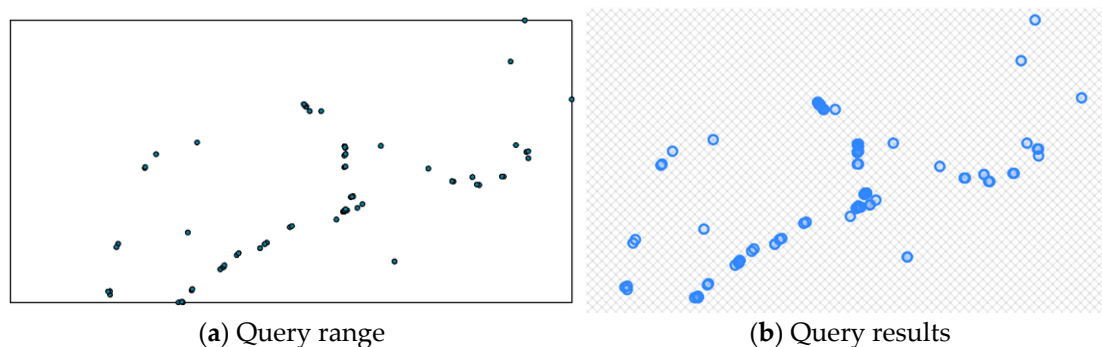


Figure 7. Query range and query results for point data.

Table 4. Query process information of *waterways*.

Query Size	Our Scheme		Comparison Scheme/ms	TU Decrypt & Filter/ms	Communication Cost/KB	Efficiency Loss/%
	Query Range Encrypt/ms	CS Query/ms				
1%	0.17	0.96	0.93	24.33	63.11	−0.79
5%	0.17	3.85	4.07	115.62	333.41	0.04
10%	0.17	7.27	7.34	217.87	652.50	−0.04
15%	0.17	11.06	10.91	343.46	1022.57	−0.09

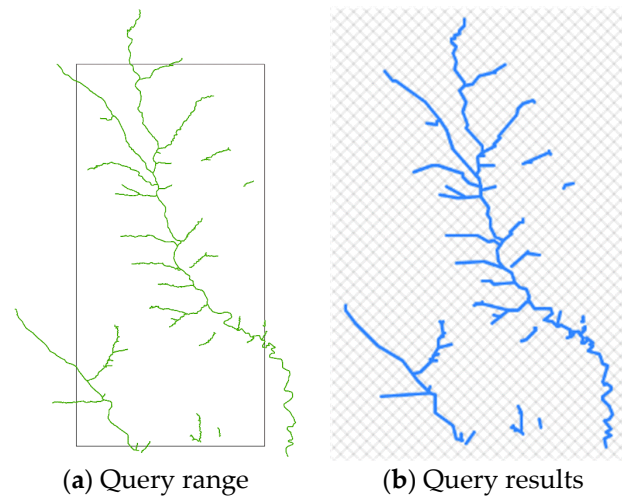


Figure 8. Query range and query results for polyline data.

Table 5. Query process information of buildings.

Query Size	Our Scheme		Comparison Scheme/ms	TU Decrypt & Filter/ms	Communication Cost/KB	Efficiency Loss/%
	Query Range En-crypt/ms	CS Query/ms				
1%	0.17	6.83	7.23	84.82	225.26	0.25
5%	0.17	37.82	36.41	392.41	1114.41	−0.37
10%	0.17	80.31	77.45	747.12	2128.73	−0.37
15%	0.17	126.81	122.86	1541.18	4559.37	−0.25

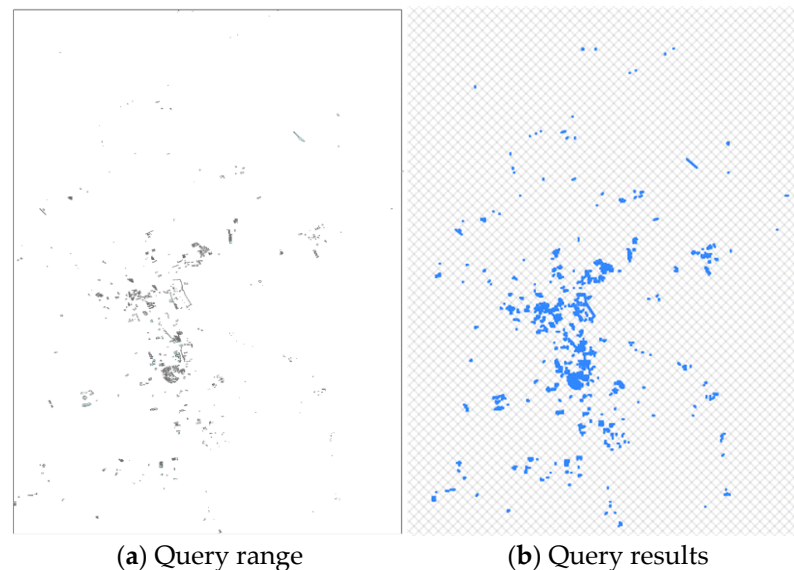


Figure 9. Query range and query results for polygon data.

Table 6. Summary of experimental results on remaining datasets.

Datasets	Query Size	Our Scheme		Comparison Scheme/ms	TU Decrypt & Filter/ms	Communication Cost/KB	Efficiency Loss/%
		Query Range En-crypt/ms	CS Query/ms				
<i>places</i>	15%	0.17	3.60	3.56	16.58	41.06	−1.04
<i>traffic</i>	15%	0.17	9.39	9.41	40.64	112.54	−0.30
<i>waterways</i>	15%	0.17	11.06	10.91	343.46	1022.57	−0.09
<i>railways</i>	15%	0.17	8.46	8.22	107.82	311.87	−0.35

<i>roads</i>	15%	0.17	166.28	168.39	2663.21	8681.05	0.07
<i>pois</i>	15%	0.17	4.84	4.89	90.72	255.51	−0.13
<i>landuse</i>	15%	0.17	90.52	91.02	2338.46	7395.86	0.01
<i>buildings</i>	15%	0.17	126.81	122.86	1541.18	4559.37	−0.25

7.3. Point Data Query Comparison

To ensure a fair comparison, we select the DISC [10] as the comparison scheme for point data query. Our scheme and DISC both utilize R-tree and OPE, thereby enabling a comparative performance analysis of queries based on MBR and SFC code within a similar technical framework. The experimental results are shown in Figures 10 and 11. In DISC, the SFC mapping causes multiple point data to share the same code, which is particularly pronounced in densely distributed regions. During the query, CS returns all point data corresponding to the matching codes as the query results to TU, resulting in a decrease in query efficiency. Figure 10 shows that as the query size increases, the time cost of the CS query and TU decryption and filtering in DISC grows rapidly, significantly higher than that of our scheme. There are two reasons for this outcome. First, the query results in DISC contain a large number of false positives, which increases the burden on data transmission and TU computation. Second, when CS performing index retrieval, the number of computations required by DISC is far greater than that of our scheme. As a result, the query efficiency of our scheme is superior to that of DISC. Figure 11 illustrates the communication costs of query results in the two schemes. As the query results in DISC contain a large number of false positives, the burden on data transmission and TU computation is aggravated.

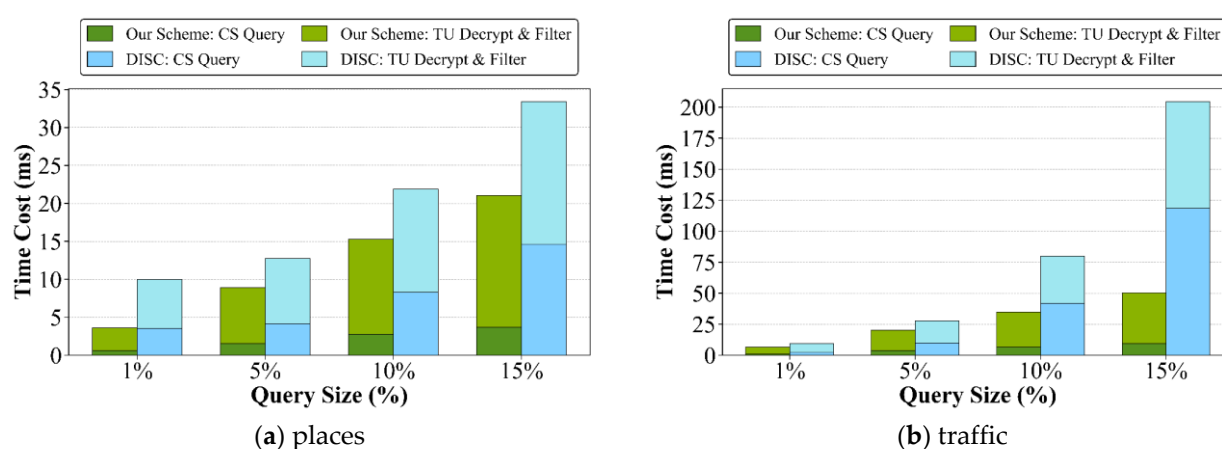


Figure 10. CS query time and TU decrypt-and-filter time for point data.

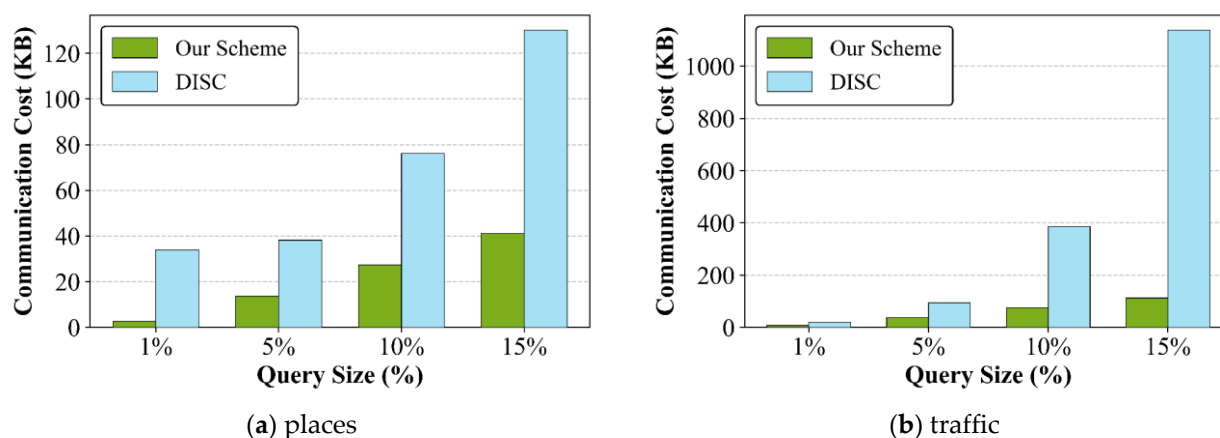


Figure 11. The communication cost for point data.

8. Conclusions

Existing spatial range query schemes for encrypted vector data are primarily designed for point data and have limited support for polyline and polygon data. Therefore, we propose a secure and efficient spatial range query scheme that supports multi-type encrypted vector data. Experimental results demonstrate that although the proposed scheme brings a certain efficiency loss compared with the comparison scheme, with a maximum efficiency loss of only 3.97%, it effectively prevents the leakage of both vector data and query ranges, thereby achieving higher security with relatively low performance overhead. In addition, we compare our scheme with DISC on point data query. The results demonstrate that our scheme achieves better performance in terms of both query efficiency and communication cost. However, for each point object, the MBR degenerates into a single point, which may reveal data type and order relations. In future work, we will investigate more privacy-enhanced index construction methods for point data that can reduce such information leakage without significantly affecting query performance. In summary, our scheme achieves a balance between data security and efficiency, providing a feasible solution for spatial range query over multi-type encrypted vector data.

Author Contributions: The authors confirm contributions to the paper as follows: Study conception and design: Q.Y.; data collection: Q.Y.; analysis and interpretation of results: Q.Y.; writing—original draft preparation: Q.Y.; writing—review and editing: Z.M., D.L., Q.Y., X.W., J.H. and H.L.; supervision: Z.M., D.L., X.W., J.H. and H.L.; funding acquisition: Z.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key Research and Development Program of China (grant number 2024YFB2505804).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data will be made available on request.

Acknowledgments: The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

Conflicts of Interest: The authors have no conflicts of interest to declare that are relevant to the content of this article. The authors have no financial interests to disclose.

References

1. Yang, Y.; Huang, X.; Liu, X.; Cheng, H.; Weng, J.; Luo, X.; Chang, V. A Comprehensive Survey on Secure Outsourced Computation and Its Applications. *IEEE Access* **2019**, *7*, 159426–159465. <https://doi.org/10.1109/ACCESS.2019.2949782>.
2. Domingo-Ferrer, J.; Farràs, O.; Ribes-González, J.; Sánchez, D. Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges. *Comput. Commun.* **2019**, *140–141*, 38–60. <https://doi.org/10.1016/j.comcom.2019.04.011>.
3. Xu, L.; Duan, H.; Zhou, A.; Yuan, X.; Wang, C. Interpreting and mitigating leakage-abuse attacks in searchable symmetric encryption. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 5310–5325. <https://doi.org/10.1109/TIFS.2021.3128823>.
4. Wang, H.; Hu, W.; Quan, Y.; Mu, D. Private Spatial Range Queries Over Outsourced Data: A Survey. In Proceedings of the 2025 IEEE International Conference on Big Data (BigData), Macau, China, 8–11 December 2025; pp. 6504–6513. <https://doi.org/10.1109/BigData66926.2025.11400818>.
5. Wu, H.; Düdder, B.; Wang, L.; Cao, Z.; Zhou, J.; Feng, X. Survey on Secure Keyword Search over Outsourced Data: From Cloud to Blockchain-assisted Architecture. *ACM Comput. Surv.* **2023**, *56*, 1–40. <https://doi.org/10.1145/3617824>.
6. Tu, X.; Bao, H.; Lu, R.; Huang, C.; Dai, H.-N. PMRK: Privacy-Preserving Multidimensional Range Query with Keyword Search Over Spatial Data. *IEEE Internet Things J.* **2024**, *11*, 10464–10478. <https://doi.org/10.1109/JIOT.2023.3326004>.
7. Fugkeaw, S.; Deevijit, J.; Ueasathitwong, P.; Thanyasukpaisal, T. EVSEB: Efficient and Verifiable Searchable Encryption with Boolean Search for Encrypted Cloud Logs. *IEEE Access* **2025**, *13*, 101177–101195. <https://doi.org/10.1109/ACCESS.2025.3577466>.

8. Rady, M.; Abdelkader, T.; Ismail, R. Integrity and Confidentiality in Cloud Outsourced Data. *Ain Shams Eng. J.* **2019**, *10*, 275–285. <https://doi.org/10.1016/j.asej.2019.03.002>.
9. Yao, X.; Li, G. Big spatial vector data management: A review. *Big Earth Data* **2018**, *2*, 108–129. <https://doi.org/10.1080/20964471.2018.1432115>.
10. Kamel, I.; Talha, A.M.; Aghbari, Z.A. Dynamic spatial index for efficient query processing on the cloud. *J. Cloud Comput.* **2017**, *6*, 5. <https://doi.org/10.1186/s13677-017-0077-0>.
11. Wang, B.; Li, M.; Wang, H. Geometric Range Search on Encrypted Spatial Data. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 704–719. <https://doi.org/10.1109/TIFS.2015.2506145>.
12. Miao, Y.; Yang, Y.; Li, X.; Liu, Z.; Li, H.; Choo, K.-K.R.; Deng, R.H. Efficient Privacy-Preserving Spatial Range Query Over Outsourced Encrypted Data. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 3921–3933. <https://doi.org/10.1109/TIFS.2023.3288453>.
13. Xu, G.; Li, H.; Dai, Y.; Yang, K.; Lin, X. Enabling Efficient and Geometric Range Query with Access Control Over Encrypted Spatial Data. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 870–885. <https://doi.org/10.1109/TIFS.2018.2868162>.
14. Wang, X.; Ma, J.; Liu, X.; Deng, R.H.; Miao, Y.; Zhu, D.; Ma, Z. Search me in the dark: Privacy-preserving boolean range query over encrypted spatial data. In *Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications*; IEEE: Toronto, ON, Canada, 2020. <https://doi.org/10.1109/INFOCOM41043.2020.9155505>.
15. Zheng, Z.; Cao, Z.; Shen, J. Practical and Secure Circular Range Search on Private Spatial Data. In *Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Guangzhou, China, 29 December 2020–1 January 2021; pp. 639–645. <https://doi.org/10.1109/TrustCom50675.2020.00090>.
16. Cui, N.; Yang, X.; Wang, B.; Geng, J.; Li, J. Secure range query over encrypted data in outsourced environments. *World Wide Web* **2019**, *23*, 491–517. <https://doi.org/10.1007/s11280-019-00726-5>.
17. Peng, Y.; Li, H.; Cui, J.; Zhu, Y.; Peng, C. An Efficient Range Query Model over Encrypted Outsourced Data Using Secure K-D Tree. In *Proceedings of the 2016 International Conference on Networking and Network Applications (NaNA)*, Hakodate, Japan, 23–25 July 2016; pp. 250–253. <https://doi.org/10.1109/NaNA.2016.31>.
18. Hamza, A.; Kumar, B. A Review Paper on DES, AES, RSA Encryption Standards. In *Proceedings of the 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, Moradabad, India, 4–5 December 2020; pp. 333–338. <https://doi.org/10.1109/SMART50582.2020.9336800>.
19. Alenezi, M.N.; Alabdulrazzaq, H.; Mohammad, N.Q. Symmetric encryption algorithms: Review and evaluation study. *Int. J. Commun. Netw. Inf. Secur.* **2020**, *12*, 256–272.
20. Yu, P.; Ni, W.; Liu, R.P.; Zhang, Z.; Zhang, H.; Wen, Q. Efficient Encrypted Range Query on Cloud Platforms. *ACM Trans. Cyber-Phys. Syst.* **2022**, *6*, 1–23. <https://doi.org/10.1145/3548657>.
21. Abdennebi, A.; Kaya, K. A bloom filter survey: Variants for different domain applications. *arXiv* **2021**, arXiv:210612189. <https://doi.org/10.48550/arXiv.2106.12189>.
22. Bader, M. *Space-Filling Curves: An Introduction with Applications in Scientific Computing*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012. <https://doi.org/10.1007/978-3-642-31046-1>.
23. Al-Mamun, A.; Wu, H.; He, Q.; Wang, J.; Aref, W.G. A Survey of Learned Indexes for the Multi-dimensional Space. *ACM Comput. Surv.* **2025**, *58*, 1–37. <https://doi.org/10.1145/3768575>.
24. Miao, Y.; Yu, J.; Li, J.; Li, X.; Feng, J.; Liu, Z.; Deng, R.H. Security-Enhanced Spatial Range Query Over Large-Scale Encrypted Mobile Cloud Datasets. *IEEE Trans. Mob. Comput.* **2026**, *25*, 1184–1199. <https://doi.org/10.1109/TMC.2025.3599519>.
25. Kim, H.-I.; Hong, S.-T.; Chang, J.-W. (Eds.) Hilbert-curve based cryptographic transformation scheme for protecting data privacy on outsourced private spatial data. In *Proceedings of the 2014 International Conference on Big Data and Smart Computing (BIGCOMP)*; IEEE: Bangkok, Thailand, 2014. <https://doi.org/10.1109/BIGCOMP.2014.6741411>.
26. Jagadish, H.V. (Ed.) Linear clustering of objects with multiple attributes. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*; Association for Computing Machinery: New York, NY, USA, 1990. <https://doi.org/10.1145/93597.98742>.
27. Li, M.; Jia, C.; Du, R.; Ha, G. Efficient Conjunctive Geometric Range Query Over Encrypted Spatial Data with Learned Index. *IEEE Trans. Comput.* **2025**, *74*, 3995–4009. <https://doi.org/10.1109/TC.2025.3604470>.
28. Wang, H.; Fu, X.; Xu, J.; Lu, H. Learned Index for Spatial Queries. In *Proceedings of the 2019 20th IEEE International Conference on Mobile Data Management (MDM)*, Hong Kong, China, 10–13 June 2019; pp. 569–574. <https://doi.org/10.1109/MDM.2019.00121>.
29. Wang, Z.; Lu, J.; Wu, J.; Tian, Y.; Song, W.; Li, Q.; Zhang, D. Towards Privacy-Preserving Range Queries with Secure Learned Spatial Index over Encrypted Data. *arXiv* **2025**, arXiv:251203669. <https://doi.org/10.48550/arXiv.2512.03669>.

30. Wang, C.; Yu, J.; Zhao, Z. GLIN: A (G)eneric (L)earned (In)dexing Mechanism for Complex Geometries. In *Proceedings of the 11th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*; Association for Computing Machinery: New York, NY, USA, 2023; pp. 1–12. <https://doi.org/10.1145/3615833.3628590>.
31. Boldyreva, A.; Chenette, N.; Lee, Y.; O’neill, A. (Eds.) Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Cologne, Germany, 2009. https://doi.org/10.1007/978-3-642-01001-9_13.
32. Li, M.; Wang, H.; Dai, H.; Li, M.; Chai, C.; Gu, R.; Chen, F.; Chen, Z.; Li, S.; Liu, Q.; et al. A Survey of Multi-Dimensional Indexes: Past and Future Trends. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 3635–3655. <https://doi.org/10.1109/TKDE.2024.3364183>.
33. Leutenegger, S.T.; Lopez, M.A.; Edgington, J. (Eds.) STR: A simple and efficient algorithm for R-tree packing. In *Proceedings of the 13th International Conference on Data Engineering*; IEEE: Birmingham, UK, 1997. <https://doi.org/10.1109/ICDE.1997.582015>.
34. Bobkina, V.; Arseniev, D.; Zhiltsova, A. (Eds.) *Coordinate Support of Geographic Information Space*; IOP Conference Series: Materials Science and Engineering; IOP Publishing: Bristol, UK, 2020. <https://doi.org/10.1088/1757-899X/913/4/042023>.
35. Tan, C.; Deng, X.; Zhang, L. Identification of block ciphers under CBC mode. *Procedia Comput. Sci.* **2018**, *131*, 65–71. <https://doi.org/10.1016/j.procs.2018.04.186>.
36. Praschl, C.; Krauss, O. Extending 3D geometric file formats for geospatial applications. *Appl. Geomat.* **2023**, *16*, 161–180. <https://doi.org/10.1007/s12518-023-00543-6>.
37. Tóth, R.; Balla, D.; Zichar, M. (Eds.) *Analysis and Optimization of KML Files*; INTED2020 Proceedings; IATED: Valencia, Spain, 2020. <https://doi.org/10.21125/inted.2020.1175>.
38. Portele, C. *Opengis® Geography Markup Language (GML) Encoding Standard*; Version 3.2.1; Open Geospatial Consortium: Arlington, VA, USA, 2007. <https://doi.org/10.25607/OBP-645>.
39. Tvrdy, P.; Long, J. *National Transportation Library Datasets File Formats Dictionary*; National Transportation Library: Washington, DC, USA, 2024. <https://doi.org/10.21949/1530074>.
40. Yusof, S.N.; Kamel Ariffin, M.R.; Lau, T.S.C.; Salim, N.R.; Yip, S.-C.; Yap, T.T.V. An IND-CPA Analysis of a Cryptosystem Based on Bivariate Polynomial Reconstruction Problem. *Axioms* **2023**, *12*, 304. <https://doi.org/10.3390/axioms12030304>.
41. Boldyreva, A.; Chenette, N.; O’Neill, A. (Eds.) Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Annual Cryptology Conference*; Springer: Santa Barbara, CA, USA, 2011. https://doi.org/10.1007/978-3-642-22792-9_33.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.