Privacy-preserving Multi-Dimensional Range Query Optimization Across Multiple Sources

Chengyue Ge[1], Yuan Shen[3,4⋆], Wei Song[1,2,5(✉)], Jinyang Ren[1], and Yuan Feng[6]

[1] School of Computer Science, Wuhan University, Wuhan, China
{chengyuege,songwei,renjinyang}@whu.edu.cn
[2] Xinjiang Production and Construction Corps Key Laboratory of Computing Intelligence and Network Information Security, Shihezi University, Shihezi, China
[3] School of Software, Pingdingshan University, Pingdingshan, China
shen@pdsu.edu.cn
[4] Henan International Joint Laboratory for Multidimensional Topology and Carcinogenic Characteristics Analysis of Atmospheric Particulate Matter PM2.5, Pingdingshan, China
[5] Intellectual Computing Laboratory for Cultural Heritage, Wuhan University, Wuhan, China
[6] Wuhan Dameng Database Company Limited, Wuhan, China
fengyuan@dameng.com

Abstract. Multi-dimensional range queries play a crucial role in various collaborative data analysis tasks, such as healthcare insurance, retail customer observation, and financial risk management. These queries across different platforms enable users to obtain precise and comprehensive research data, which cannot be achieved with single-table queries. To protect privacy, many researchers have focused on designing privacy-preserving range query schemes. However, when these methods are applied to multiple tables, existing solutions face security and efficiency issues. The security concern involves the exposure of intermediate results in each table, which can reveal whether the information of a research object meets the query range of each table. The efficiency problem arises from building an index on large-scale data for each table and the requirement to scan each table thoroughly. To address these problems, this paper proposes a privacy-preserving collaborative multi-dimensional range query scheme that balances efficiency and security in multiple sources scenarios. We first introduce a secure construction scheme for a collaborative R-tree among multiple sources. Additionally, we propose a pruning strategy based on dimension sensitivity to accelerate secure multi-dimensional range queries. Theoretical analysis and experimental results demonstrate the security and effectiveness of our method.

Keywords: Privacy-preserving Range Query · Query Optimization · Data Security.

⋆ First Author and Second Author contribute equally to this work.

1 Introduction

Multi-dimensional range query plays a crucial role in collaborative data analysis such as healthcare insurance, retail customer observation, and financial risk management[2,20]. For better statistics, data users often filter out the target objects with range requirements on several attributes across multiple tables from different sources or databases. For instance, to analyze the consuming behavior of the social media user, the query can be $followers \in [1000, 10000] \wedge posts \in [100, 150]$ in social media application and $orders\_num \in [10, 20] \wedge total\_amount \in [2000, 5000]$ in online shopping services. However, the cloud server is not completely trusted and tends to infer the users' privacy. It is important to provide an efficient privacy-preserving multi-dimensional range query scheme across multiple sources.

There are many privacy-preserving range query schemes[19,17,4], including multi-dimensional range queries[18,21]. These schemes first encode the data to support secure comparison, using various encryption schemes such as Order-Revealing Encryption (ORE)[3]. Numerous efficient index structures, such as tree-based schemes[13,1] or bucketization[7], are applied to speed up the range query process. For multi-dimensional range queries, many studies[21,18] transform the secure comparison between data and bounds into point intersection prediction and range intersection prediction. Existing work mainly focuses on secure range query processes in a single table.

Table 1: Query time comparison for different strategies. Experiments are conducted with a data volume of 100,000 and 5 dimensions. STF Query refers to the Small Table First Query strategy.

Query Strategy	Query Time (ms)	Execution Method	Intermediate Result
Collaborative Index	0.1407	Join first, then query	No
Cross Query	346.2601	Query first, then join	Yes
STF Query	9.4025	Query small table first	Yes
Linear Query	420.7842	Sequential query	Yes

However, when these methods are applied to multiple tables, two main issues arise: (1) Single-table privacy leakage. Single-table privacy reveals which records satisfy a range query on a single table. In multi-table scenarios, attackers can combine intermediate results to infer patterns, reconstruct data, and exploit attribute relationships, significantly amplifying privacy risks. (2) Efficiency. As shown in Table 1, our experiments on plaintext indicate that a collaborative index that consolidates attributes from all tables into one index significantly accelerates query performance compared to querying each table individually. The existing work needs to query on each table, resulting in huge costs for the unnecessary secure computation on the intermediate result. Although there are many studies solving secure collaborative data analysis, they mainly concentrate

on higher-level operations, such as aggregation and grouping. To sum up, there still lacks a practical collaborative index to support privacy-preserving multi-dimensional range query over multiple tables.

Motivated by these, our goal is to improve the performance of privacy-preserving multi-dimensional range query schemes in multiple-table scenarios. We propose a collaborative R-tree that leverages the efficiency of R-trees while ensuring security in cross-platform environments. The key features of our proposed collaborative and secure range query scheme include: (i) batching the data and precomputing the incremental costs before insertion; (ii) utilizing a secure comparison protocol based on homomorphic encryption; (iii) implementing a secure encoding scheme for multiple data owners, inspired by the PRQ model[21]; and (iv) employing a pruning strategy that considers dimension sensitivity.

The main contributions of this paper can be summarized as:

- First, to protect privacy during index construction, we propose a protocol to securely build a collaborative R-tree using homomorphic encryption-based secure comparison protocols.
- Second, to support multi-dimensional range queries across multiple data owners, we propose a secure framework using multiple keys. It is implemented through a secure encoding scheme based on the PRQ method.
- Third, to speed up the query process, we introduce a pruning strategy based on dimension sensitivity. We design a secure computation protocol to obtain dimension sensitivity.
- Lastly, we conduct comprehensive theoretical analyses and extensive experiments to demonstrate the performance of our work.

2 Related Work

To address privacy-preserving range query problem, researchers have proposed various schemes, which can be categorized into three main groups.

Secure Comparison in Range Query: Guo *et al.*[6] used ORE to achieve privacy-preserving geographic range queries. SOREL[19] enhanced security beyond bit-based ORE schemes by adding prefixes and suffixes to the encoding. Lv *et al.*[11] proposed m-ORE to enable range queries across multiple users. Other public key cryptography methods also support secure data comparison. For example, EPRQ [10] used hidden vector encryption to enable data comparison.

Indexing for Range Query: Lee *et al.*[7] defined an encryption scheme over ordered bucketization. However, bucketization schemes often suffer from the false positive problem. Tree-based indexes can reduce query complexity to sub-linear levels. Tang[16] *et al.* adopted tree-based attribute structure to support range query in healthcare field. Wang *et al.*[17] used a quadtree to support Boolean range queries on encrypted spatial data. Chang *et al.*[4] proposed an efficient oblivious query processing method for range and k-nearest neighbor (kNN) queries, building an oblivious tree structure (B-tree/R-tree).

Privacy-preserving Multi-dimensional Range Query: In multidimensional range query scenario, Shi *et al.*[14] first proposed MRQED. They represent data intervals using binary interval trees. However, it must create a tree for every dimension, which leads to practicality issues. Building on the R-tree, TRQED[18] divided the problem into point and range intersection predicate encryption. PRQ[21] improved the intersection algorithms by adapting the data comparison algorithms to a lightweight matrix representation. However, these R-tree-based methods must evaluate all dimensions, resulting in significant overhead when dealing with large-scale dimensions.

In conclusion, there is a strong need for a collaborative and secure method for range queries, as it ensures efficient and reliable multi-dimensional data retrieval from multiple sources.

3 Prelminaries

3.1 Homomorphic Encryption

Homomorphic encryption allows operations to be performed on ciphertexts while preserving the privacy of the underlying data. The definition is as follows: Given n plaintexts $m_1, m_2, ..., m_n$, an encryption algorithm E, a decryption algorithm D, and a computable function f, the scheme satisfies $D(f([\![m_1]\!], [\![m_2]\!], ..., [\![m_n]\!])) = f(m_1, m_2, ..., m_n)$, where $[\![m]\!]$ denotes the ciphertext after the encryption. There are various cryptographic algorithms supporting homomorphic encryption. For instance, the Paillier[12] encryption scheme supports both additive and multiplicative homomorphism as follows:

$$[\![m]\!] \times [\![n]\!] = [\![m + n]\!]$$

$$[\![m]\!]^a = [\![a \times m]\!]$$

3.2 PRQ Scheme

The PRQ scheme [21] includes two encryption methods: multi-dimensional point intersection predicate encryption (PIPE) and range intersection predicate encryption (RIPE). In PIPE, the secret key for matrix encryption is denoted as $sk_P = \{\tilde{\mathbf{M}}, \tilde{\mathbf{M}}'\}$. Data points are encrypted as $\{\mathrm{CP}_{i,1}, \mathrm{CP}_{i,2}\}_{i=1}^{d}$, where d indicates the dimension index. In RIPE, the secret key is represented $sk_R = \{\bar{\mathbf{M}}, \bar{\mathbf{M}}'\}$. Here, the lower and upper bounds of the minimum bounding rectangle (MBR) $\mathbf{B}$ are encrypted as $\{\mathrm{CB}_{l,i,1}, \mathrm{CB}_{l,i,2}, \mathrm{CB}_{u,i,1}, \mathrm{CB}_{u,i,2}\}_{i=1}^{d}$. With these keys, the token of the query $\mathbf{Q}$ is encrypted as $\{\mathrm{PT}_i\}_{i=1}^{d}$ for PIPE and $\{\mathrm{RT}_{l,i}, \mathrm{RT}_{u,i}\}_{i=1}^{d}$ for RIPE. To determine whether a data point $\mathbf{x}$ is included in the query, or if an MBR intersects with the query, the cloud server only needs to evaluate the equation as below:

$$\mathbf{x} \in \mathbf{Q} \Leftrightarrow \sum_{i=1}^{d} (\mathrm{CP}_{i,1} \mathrm{PT}_i \mathrm{CP}_{i,2}) = 0$$

$$\mathbf{B} \cap \mathbf{Q} \neq \emptyset \Leftrightarrow \sum_{i=1}^{d} (\mathrm{CB}_{l,i,1}\mathrm{RT}_{u,i}\mathrm{CB}_{l,i,2} + \mathrm{CB}_{u,i,1}\mathrm{RT}_{l,i}\mathrm{CB}_{u,i,2}) = 0$$

4 System Overview

4.1 System Model

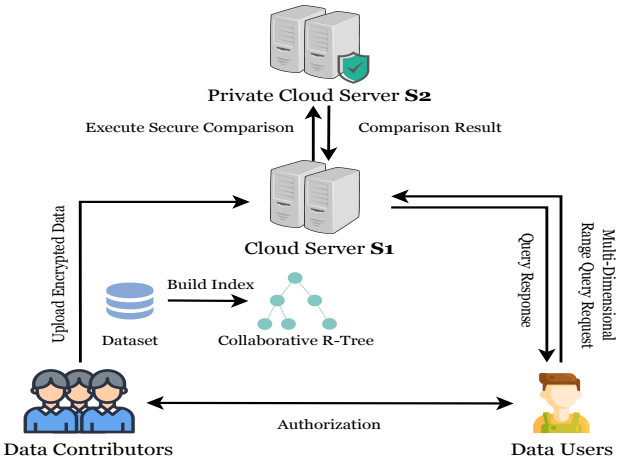


Fig. 1: System Model

In our paper, the system consists of three main roles: the cloud server, the data user, and the data owner, as illustrated in Fig. 1.

- Data Contributors: Contributors encrypt recorded objects, with cross-table common subsets serving as candidate research items. Assuming aligned tables (inspired by [5]), achieved via Private Set Intersection (PSI) protocols (outside our scope), data contributors collaboratively construct an R-tree based on dimension combinations, such as $(followers, likes, amount, price)$ in the context of analyzing social media users' online shopping behavior.
- Data Users: Data users are researchers using multi-dimensional range query to filter out valid research objects. Upon receiving the appropriate authorization keys, they can utilize the range query service to perform analysis.
- Cloud Server $\mathbf{S}_1$: $\mathbf{S}_1$ is a public cloud server which is responsible for storing encrypted data and maintaining the index structure. It performs searches on the encrypted data to identify records that meet the range of the queries, and subsequently returns the final results to the data user.
- Private Cloud Server $\mathbf{S}_2$: $\mathbf{S}_2$ is a private cloud server that data owners trust to execute and accelerate secure comparisons with $\mathbf{S}_1$. The two cloud servers operate in hybrid cloud mode and do not collude.

4.2 Data Representation

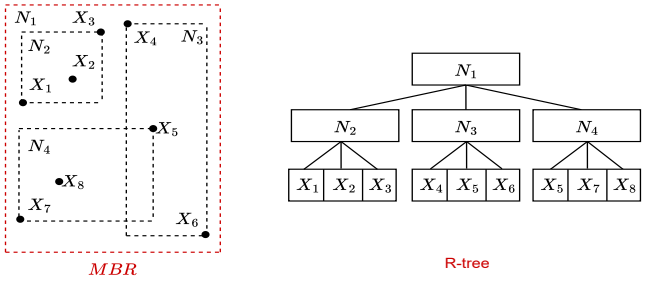


Fig. 2: Data Representation

The data representation in our work is shown in Fig.2:

Data Point ($\mathcal{X}$): Each record in the table is represented by $\mathcal{X} = (x_1, x_2, \ldots, x_d)$, where d is the number of dimensions and x_i is the attribute value of the i-th dimension.

Minimum Bounding Rectangle (MBR): In the R-tree, an MBR defines the smallest rectangle that contains a set of data points or child nodes. It is represented as $MBR = [b_{l,1}, b_{u,1}] \times [b_{l,2}, b_{u,2}] \times \cdots \times [b_{l,d}, b_{u,d}]$, where $[b_{l,i}, b_{u,i}]$ represents the range of the i-th dimension.

Cost of MBR ($C(MBR)$): The cost of an MBR is used to evaluate the spatial extent of all data points within a node. It is calculated as the equation below, which represents the perimeter of the bounding rectangle. The cost of ith dimension is $C(MBR)_i = b_{u,i} - b_{l,i}$.

$$C(MBR) = (b_{u,1} - b_{l,1}) + (b_{u,2} - b_{l,2}) + \cdots + (b_{u,d} - b_{l,d})$$

Node (N): Each node is associated with an MBR that quantifies its spatial range. Nodes have a capacity limit that determines the maximum number MAX of children (either MBRs or data points) they can hold.

4.3 Security Model

We assume that data owners are trusted, as they all benefit from collaborative data analysis, such as improved business decisions or enhanced user experience. $\mathbf{S}_1$ has powerful storage and computing capabilities. However, it is not completely trusted since it follows the protocol but exhibits curiosity towards specific private information. We aim to protect three categories of privacy:

- Data Privacy: Data is encrypted using symmetric encryption, ensuring that only authorized data users can decrypt it with the secret key provided by the data owner.

- Single-table Privacy: The collaborative index structure is designed to protect the query results of each individual table.
- Query Privacy: Query requests are transformed into trapdoors, and the query results are encrypted to maintain privacy.

4.4 Design Goals

Our work aims to address the problem of collaborative privacy-preserving multi-dimensional range queries as described above. The design goals of our approach are as follows:

1. **Privacy preservation**. We protect the data privacy, query privacy, index privacy, and single table privacy.
2. **Query optimization**. We focus on optimizing existing multi-dimensional range query encryption methods to improve the query efficiency while ensuring data security.
3. **Efficient index construction**. The collaborative-style index construction process is designed to minimize communication and computation costs.

5 Secure Collaborative Construction of R-tree

5.1 Generation of the Batching

Batching multiple data points for insertion into an R-tree significantly reduces the computational cost compared to inserting each data point individually, as it avoids recalculating the Minimum Bounding Rectangle (MBR) cost after every single insertion, especially in a multi-party setting. To prevent the large MBR expansion that could occur if distant data points are grouped into the same batch, we propose the Collaborative Encrypted Clustering Batch (CECB) Algorithm. This algorithm consists of three key steps: local clustering, global clustering, and batch generation.

Local Clustering: Each data owner is responsible for clustering based on the local dimensions using a clustering algorithm, such as K-Means. The output consists of local clustering labels $\{C_{X,1}, C_{X,2}, \ldots, C_{X,k}\}$, where $C_{X,i}$ represents the ith cluster label for party X. Each data owner then appends a random value r_X to each label to generate a randomized label, denoted as $L(C_{X,i}) = (C_{X,i}, r_X)$. These randomized labels are then sent to the cloud server.

Global Clustering: Upon receiving the randomized labels from all data owners, the cloud server combines them to produce the global clustering labels, denoted as $G = (C_{A,i} \| C_{B,j} \| \ldots)$, where $\|$ represents concatenation. The global clustering labels are then sent back to each party.

Batch Generation: Based on the global clustering labels, each data owner groups the data records that share the same global label into batches for insertion.

In this way, the data distribution of each party remains protected. The CECB algorithm ensures that each party obtains the global clustering labels without revealing the clustering details of other dimensions.

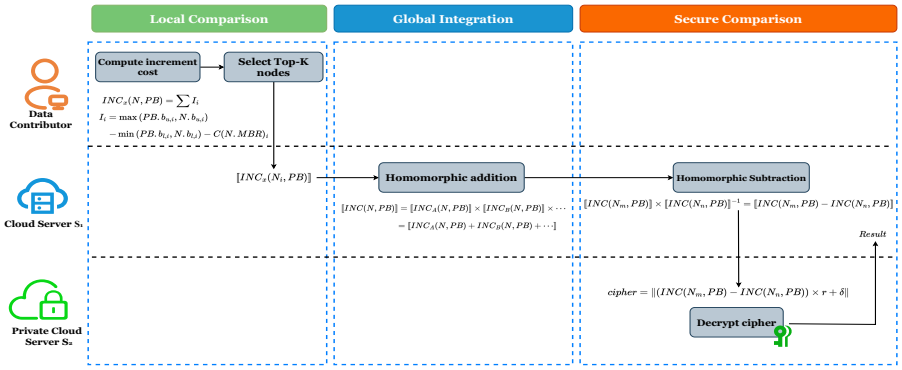


Fig. 3: Choose Node Workflow

5.2 Collaborative Data Batching Insertion

In an R-tree, the insertion workflow is illustrated in Algorithm 1. The insertion algorithm recursively finds the subtree with the lowest increment cost(lines 8-10) until it reaches a leaf node. If a node contains more than MAX data points, the split() function is called to split and adjust the tree (lines 2-7).

Algorithm 1: R-tree Insertion Algorithm insert()

Input: Root R, Data Points Batching PB
Output: Updated R-tree

```
1  u ⟵ R
2  if u is a leaf node then
3  |   Add PB to u;
4  |   if u overflows then
5  |   |   split(u); /* u has more than the allowed MAX entries */
6  |   end
7  end
8  else
9  |   v ← choose-node(u, PB);/* Determine the node under u */
10 |   insert(v, PB);/*Recursively insert*/
11 end
```

Choose Node The key step is to identify the node N with the minimum cost increment, denoted as $\mathsf{INC}(N, PB)$. Here, we use the perimeter increment to represent the cost increment, as a rectangle with a smaller perimeter typically has a smaller area [15]. The intuitive approach is to calculate the cost increment for each node and select the one with the minimum value. However, in

our multi-party setting, this results in significant communication overhead due to the need to integrate costs across tables and perform comparisons. Therefore, we propose a Collaborative Top-k Node Selection method. The entire workflow involves three steps: **local comparison, global integration, and secure comparison protocol**, as illustrated in Fig. 3.

1. **Local Comparison:** Given a batch of data points PB, each party computes the cost increment for their respective dimension as $\mathsf{INC}_x(N, PB) = \sum I_i$, where $I_i = \max(PB.b_{u,i}, N.b_{u,i}) - \min(PB.b_{l,i}, N.b_{l,i}) - C(N.MBR)_i$. Each party then sorts the cost increments to select the top-k nodes with the minimum costs, adding them to the set of candidate nodes.
2. **Global Integration:** Each party uses a shared public key and random noise to encrypt the perturbed cost increment as $[\![\mathsf{INC}_x(N_i, PB)]\!]$. The cloud server performs homomorphic addition to compute the overall cost increment for node N, represented as $[\![\mathsf{INC}(N, PB)]\!] = [\![\mathsf{INC}_A(N, PB)]\!] \times [\![\mathsf{INC}_B(N, PB)]\!] \times \cdots = [\![\mathsf{INC}_A(N, PB) + \mathsf{INC}_B(N, PB) + \dots]\!]$.
3. **Secure Comparison Protocol**: Given two nodes N_m and N_n, the cloud server first computes $[\![\mathsf{INC}(N_m, PB) - \mathsf{INC}(N_n, PB)]\!] = [\![\mathsf{INC}(N_m, PB)]\!] \times [\![\mathsf{INC}(N_n, PB)]\!]^{-1}$. Next, a large random number r and a random noise δ are selected to produce the final result as shown below. This approach preserves the sign of the subtraction while keeping the exact value hidden. The ciphertext is sent to $\mathbf{S}_2$, which decrypts it and returns 1 for a positive result or 0 for a negative one. The perturbation by the data owner ensures $\mathbf{S}_2$ cannot infer the actual value.

$$\begin{aligned} cipher &= [\![\mathsf{INC}(N_m, PB) - \mathsf{INC}(N_n, PB)]\!]^r \times [\![\delta]\!] \\ &= [\![(\mathsf{INC}(N_m, PB) - \mathsf{INC}(N_n, PB)) * r + \delta]\!] \end{aligned}$$

Split Node If the number of entities exceeds MAX, the $\mathsf{split}()$ function divides nodes using a greedy strategy: (1) identify the longest dimension in the MBR, (2) calculate the median value for children along this dimension, and (3) sort and split the children into two groups. Compared with other strategies, this method minimizes overlap and empty space, and its low computational and communication costs make it suitable for multi-party settings. Each party determines the longest dimension using our secure comparison protocol, while the responsible party calculates the median and creates two new nodes.

5.3 Other Operations

Delete: To delete a data point, the cloud server locates the leaf node (via Section 6) and updates the MBR from the leaf to the root with assistance from each party, adjusting dimensions they manage. If entities fall below the minimum threshold, the remaining points are merged into other nodes with minimal cost increment, similar to insertion.

Update: Updating a data point involves adjusting the MBR from the affected leaf node to the root, ensuring boundaries are maintained for efficient queries, following a process akin to deletion.

6 Dimension-Sensitive Multi-dimensional Range Query Scheme

The search query consists of two steps (Algorithm 2): (1) Filtration (lines 2-8): check range intersections between the query and each node's MBR from root to leaf, and (2) Verification (lines 9-15): verify if data points fall within the query range based on leaf node overlap. To ensure privacy, both the MBR and query ranges are encrypted. We extend the PRQ method [21] to the multi-party setting and accelerate the query process using dimension sensitivity. In the next part, i denotes the ith dimension, j represents the jth party and m denotes the number of the data owners.

Algorithm 2: Recursive R-tree Query Algorithm

Input: N: Current node, Q: Query range
Output: $\mathcal{R}$: Objects intersecting with Q

```
1  R ← ∅;
2  if N is not a leaf node then
3  |  if B_N ∩ Q then
4  |  |  foreach node ∈ N.child do
5  |  |  |  Query(node, Q);/* Filtration */
6  |  |  end
7  |  end
8  end
9  else
10 |  foreach X ∈ N do
11 |  |  if X ∈ Q then
12 |  |  |  R ← R ∪ {X}; /* Verification */
13 |  |  end
14 |  end
15 end
16 return R;
```

6.1 PRQ-based scheme for multiple sources

The secure multi-dimensional range query scheme for multiple sources referred to as PRQS, includes the following components: Π_{PRQS} = {KeyGen, Encrypt, TokenGen, Query}, which can be defined as follows:

- KeyGen$(N, \{\kappa_j\}_{j=1}^{m})$: For m data owners, each with a security parameter k, the data owner generates the secret keys for point intersection, range intersection, and AES encryption for a data point:

$$sk_{P,j} = \{\tilde{\mathbf{M}}, \tilde{\mathbf{M}}'\}, sk_{R,j} = \{\bar{\mathbf{M}}, \bar{\mathbf{M}}'\}, \kappa \in \{0,1\}^{\lambda}$$

- Encrypt(data,sk_P, sk_R): After aligning the data, each party is responsible for encrypting their respective data or MBR. For each MBR in the R-tree, party j calls: RipeEnc$(MBR, sk_{R,j}) \longrightarrow \{\mathrm{CB}_{l,i,1}, \mathrm{CB}_{l,i,2}, \mathrm{CB}_{u,i,1}, \mathrm{CB}_{u,i,2}\}_{i=1}^{d}$. For data points in the leaf nodes, it calls: PipeEnc$(P, sk_{p,j}) \longrightarrow \{\mathrm{CP}_{i,1}, \mathrm{CP}_{i,2}\}_{i=1}^{d}$.
- TokenGen$(q, \{sk_{p,j}^{-1}\}_{j=1}^{m}, \{sk_{r,j}^{-1}\}_{j=1}^{m})$: The data user holds the inverse secret key from every party, specifically $(\tilde{\mathbf{M}}_j^{-1}, \tilde{\mathbf{M}}_j'^{-1})_{j=1}^{m}$ for point intersection predication and $(\bar{\mathbf{M}}_j^{-1}, \bar{\mathbf{M}}_j'^{-1})_{j=1}^{m}$ for range intersection predication. It generates the query token for each party as: $T = (\{\mathrm{PT}_i\}_{i=1}^{d}, \{\mathrm{RT}_{l,i}, \mathrm{RT}_{u,i}\}_{i=1}^{d})$. Based on dimension sensitivity, it determines the checking order of the dimensions, denoted as ν.
- Query$(T, Rtree, \nu)$: Using the dimension sensitivity-based order ν, the cloud server executes the range query on the R-tree. To verify whether $MBR \cap Q = \emptyset$, it checks if $(\mathrm{CB}_{l,i,1}\mathrm{RT}_{u,i}\mathrm{CB}_{l,i,2} + \mathrm{CB}_{u,i,1}\mathrm{RT}_{l,i}\mathrm{CB}_{u,i,2} = 0)$. To check whether $P \in Q$, the condition is $\mathrm{CP}_{i,1}\mathrm{PT}_i\mathrm{CP}_{i,2} = 0$. If any dimension fails to satisfy these criteria, there is no need to continue checking the remaining dimensions.

6.2 Dimension Sensitivity-Based Query Filtration

In PRQ and other multi-dimensional encryption methods, determining whether a data point falls within a query range or intersects with the minimum bounding rectangle requires checking all dimensions using complex encoding techniques. We propose a secure pruning method based on dimension sensitivity to avoid scanning all dimensions.

Dimension sensitivity: Given a query defined as $\{[q_{l,1}, q_{u,1}], [q_{l,2}, q_{u,2}], \dots\}$ and data bounds represented by $\{[B_{l,1}, B_{u,1}], [B_{l,2}, B_{u,2}], \dots\}$, we can determine the dimension sensitivity for each dimension using the formula in Eq.(10), where $len()$ indicates the length of the range.

$$s_i = \frac{len(q_i)}{len(B_i)} \tag{10}$$

Pruning Strategy: Given the checking order of the dimensions ν, if $s_{\nu_1} < s_{\nu_2} < \cdots < s_{\nu_d}$, the query cost can be significantly reduced. This pruning strategy is based on the principles of a greedy algorithm. A smaller dimension sensitivity indicates a lower probability of intersection. Therefore, by first evaluating the dimensions with smaller sensitivity, we can avoid checking dimensions that are more likely to intersect.

To support the pruning strategy, each party should upload their dimension length to the cloud server as $L_i = \frac{1}{len(B_i)+r_l}$, where the random number r_l is a shared random number. The overall workflow for obtaining the dimension order is as follows:

1. The data user uploads the ciphertext representing the length of each dimension as $\{[\![len(q_1)]\!], [\![len(q_2)]\!], \dots\}$ and add dummy dimensions to protect the real queried dimension.

2. The cloud server calculates the ciphertext for the dimension sensitivity according to Equation (11).
3. Using all the ciphertext of dimension sensitivity, the cloud server executes the secure comparison protocol referenced in Section 5 to sort the dimension sensitivity.

$$s_i.cipher = [\![len(q_i)]\!]^{L_i} = [\![\frac{len(q_i)}{len(B_i) + r_l}]\!] \tag{11}$$

7 Security and Performance Analysis

7.1 Security Analysis

The cloud server stores three types of information: encrypted data, a collaborative index, and query requests. To meet our design goal of privacy and security, our scheme offers the following guarantees:

- **Data Privacy:** The data is encrypted by the data owner using a symmetric encryption key, denoted as κ. Only authorized data users are able to decrypt the data. Thus, the cloud server has no idea about the data.
- **Index Privacy:** MBRs (internal nodes) and data points (leaf nodes) are PRQs-encrypted with PIPE/RIPE encoding for selectivity security. The cloud server only detects query range intersections without plaintext access. Homomorphic comparisons prevent inference of exact incremental costs. Although $\mathbf{S}_2$ has the secret key, it is limited to obtaining comparison results without knowing the exact differences.
- **Query Privacy:** Query requests $\{\mathrm{PT}_i\}_{i=1}^{d}$ and $\{\mathrm{RT}_{l,i}, \mathrm{RT}_{u,i}\}_{i=1}^{d}$ are designed to prevent the cloud server from accessing plaintext queries. Random matrices help obscure query patterns. The dimension order ν, with dummy dimensions, further hides dimension sensitivity. For query results, symmetric encryption of datasets ensures the cloud server cannot access the plaintext results.
- **Single-Table Privacy:** The collaborative index is constructed using data from all queried tables, and the dimension order ν is defined across all tables. Intermediate results that only satisfy a subset of the tables are not disclosed. The cloud server can only determine if a data item meets all conditions across each table, ensuring single-table privacy.

7.2 Performance Analysis

Computing Cost. The time complexity for insertion, deletion, and querying on an R-tree is $O(\log |X|)$, where $|X|$ is the data volume. Each party incurs a cost of $T_{\mathrm{sort}} + kT_{\mathrm{encP}}$ during node selection, with T_{sort} for sorting and T_{encP} for Paillier encryption. The cloud server's cost is $cT_{\mathrm{addP}} + c^2(T_{\mathrm{mulP}} + T_{\mathrm{addP}})$, where c is the number of candidate nodes, T_{addP} and T_{mulP} are the time cost for homomorphic addition and multiplication, respectively. In the query process, the cloud server's

time cost is $N_{\text{inter}}T_{\text{RIPE}} + N_{\text{leaf}}T_{\text{PIPE}}$, where N_{inter} and N_{leaf} are the number of checked internal and leaf nodes, and T_{RIPE} and T_{PIPE} represent the time costs for range and point intersection prediction, respectively. All dimensions of the query do not need to be checked for these predictions.

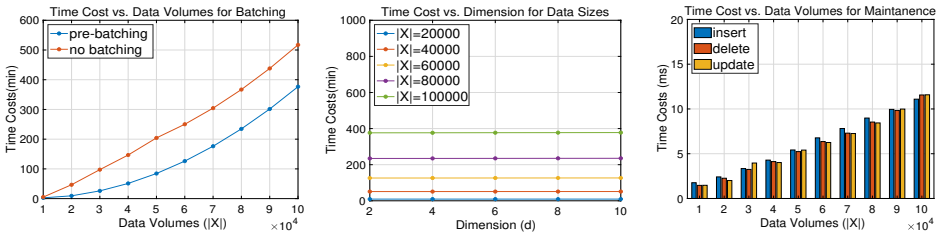
Overheads of Communication. The communication costs are primarily incurred during the index-building process, particularly during the selection of qualified subnodes. Each party's communication cost for choosing a qualified subnode is kS_{cipher}, where S_{cipher} represents the storage size of the Paillier encryption. In the query process, the size of the trapdoor is calculated as $S_{\text{PIPE}} + S_{\text{RIPE}} + dS_{cipher}$. Here, S_{PIPE} denotes the storage size of the set $\{\text{PT}_i\}_{i=1}^{d}$, while S_{RIPE} denotes the storage size of the set $\{\text{RT}_{l,i}, \text{RT}_{u,i}\}_{i=1}^{d}$.

8 Experimental Study

8.1 Experimental Setup

We implemented our scheme in Java and conducted experiments on a machine with an Apple M2 processor (@3.2GHz), 16GB of RAM, and the macOS Sonoma 14.0 operating system. The dataset we used is part of a large eCommerce store's data from 2019, collected by Open CDP (https://rees46.com/en/open-cdp), along with social media user information. The default size of the secret key in the Paillier encryption system is 1024 bits. The maximum size and minimal size of the child nodes in R-tree are set as 8 and 2. Since there is currently no solution that addresses secure multidimensional range queries across multiple tables, we chose related studies [18,21,8,9], as the basis for comparison.

8.2 Index Building Costs



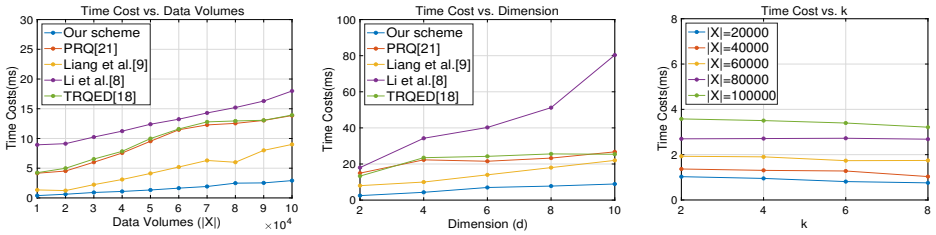
(a) Build Time vs. |X| (b) Build Time vs. d (c) Maintanence Time

Fig. 4: Evaluation for the index maintanence

We evaluate the construction time by considering data volumes, dimensions, and the operations of insertion, deletion, and updating. Every experiment is executed 1,000 times, and we calculate the average performance as the final result.

First, we compare the time cost of constructing the collaborative R-tree index using pre-batching and non-batching methods, with varying data volumes |X| ranging from 10,000 to 100,000. As shown in Fig. 4(a), the time costs for the pre-batching method are faster than those of the non-batching method, particularly with larger datasets. The pre-batching method improves performance by up to 43.81%. In Fig. 4(b), we measure the construction time with different numbers of dimensions. The results indicate that construction time remains stable across multiple dimensions, as the primary influence on the tree's structure is the data volume. Additionally, the time cost for encrypting the MBR is quite low, averaging 0.8 milliseconds per encryption. Regarding other operation time costs—such as insertion, updating, and deletion, Fig. 4(c) demonstrates that these operations have minor differences in time costs since they follow similar processes.

8.3 Time Costs of Range Query



(a) Query Time vs. |X| (b) Query Time vs. d (c) Query Time vs. k

Fig. 5: Evaluation for the Range Query

We assess the query time by considering data volumes, dimensions, and the number of selected nodes k. The related schemes [21,18] are designed for multi-dimensional range queries over a single table, while other schemes [9,8] focus on single-dimensional range queries across multiple sources. First, we evaluate the range query time across different data volumes, ranging from 10,000 to 100,000 with $d = 3$, as illustrated in Fig. 5(a). Our scheme, which employs a pruning strategy, shows a lower cost compared to multidimensional range query schemes [21,18], resulting in a performance improvement of 89.7%. In Fig. 5(b), we examine the impact of increasing dimensions on query time costs. The results indicate that our scheme remains stable as the number of dimensions increases, since the pruning strategy does not require checking all dimensions. Finally, we evaluate how the number of selected nodes k, which each party sets as candidate nodes with minimum increment costs, affects the final query time costs. As shown in Fig. 5(c), the time costs remain stable as k increases, indicating that a small number of candidate nodes can still achieve low time costs.

8.4 Communication cost

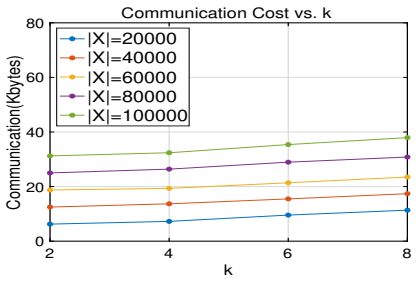


Fig. 6: Communication costs vs. k

To evaluate communication costs between the cloud server and the data owners during index establishment, we assess the impact of the variable k and the data volumes $|X|$ on these costs. The results are illustrated in Fig. 6. It is evident that the primary influence on communication costs comes from data volumes, which lead to an increase in the number of node selection operations. In contrast, the number of selected nodes k has a minimal effect on costs since there is a significant overlap between the candidate nodes of each party.

9 Conclusion

In this study, we find out the challenges associated with privacy-preserving multi-dimensional range queries across multiple tables from different sources. To address these challenges, we propose a collaborative R-tree-based indexing method that operates under secure transmission protocols. Additionally, we present a secure framework for multi-dimensional range queries, aimed at accelerating the query process based on dimension sensitivity. In conclusion, our work achieves efficiency and security in the privacy-preserving multidimensional range query in the multiple sources scenarios.

Acknowledgments. This work is supported by the National Natural Science Foundation of China Nos. 62372340 and 62072349, the Major Technical Research Project of Hubei Province No. 2023BAA018, Key Research and Development Program Projects of the Corps Nos. 2024AA010, 2024AB080, Ali cooperation research and development project No. 24566207, and Xinjiang Cyber Information Technology Innovation Research Project No. 12421607. Xinjiang Production and Construction Corps Key Laboratory of Computing Intelligence and Network Information Security Open Fund No. CZ002702-02.

References

1. Adel'son-Vel'skii, G.M.: An algorithm for the organization of information. Soviet Math. pp. 1259–1263 (1962)
2. Bater, J., Elliott, G., Eggen, C., Goel, S., Kho, A.N., Rogers, J.: SMCQL: secure query processing for private data networks. VLDB Endow. pp. 673–684 (2017)
3. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In: ASIACRYPT. pp. 563–594 (2015)
4. Chang, Z., Xie, D., Li, F., Phillips, J.M., Balasubramonian, R.: Efficient oblivious query processing for range and knn queries. TKDE pp. 5741–5754 (2021)
5. Fang, W., Cao, S., Hua, G., Ma, J., Yu, Y., Huang, Q., Feng, J., Tan, J., Zan, X., Duan, P., Yang, Y., Wang, L., Zhang, K., Wang, L.: Secretflow-scql: A secure collaborative query platform. Proc. VLDB Endow. pp. 3987–4000 (2024)
6. Guo, Y., Xie, H., Wang, C., Jia, X.: Enabling privacy-preserving geographic range query in fog-enhanced iot services. TDSC pp. 3401–3416 (2022)
7. Lee, Y.: Secure ordered bucketization. TDSC pp. 292–303 (2014)
8. Li, Y., Liu, W., Zhu, Y., Chen, H., Cheng, H., Chen, T., Hu, P., Huan, R.: Privacy-aware fuzzy range query processing over distributed edge devices. TFS pp. 1421–1435 (2022)
9. Liang, J., Qin, Z., Xiao, S., Zhang, J., Yin, H., Li, K.: Privacy-preserving range query over multi-source electronic health records in public clouds. JPDC pp. 127–139 (2020)
10. Liang, Y., Ma, J., Miao, Y., Su, Y., Deng, R.H.: Efficient and privacy-preserving encode-based range query over encrypted cloud data. TIFS pp. 9085–9099 (2024)
11. Lv, C., Wang, J., Sun, S.F., Wang, Y., Qi, S., Chen, X.: Efficient multi-client order-revealing encryption and its applications. In: ESORICS. pp. 44–63 (2021)
12. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology - EUROCRYPT. pp. 223–238 (1999)
13. Procopiuc, O., Agarwal, P.K., Arge, L., Vitter, J.S.: Bkd-tree: A dynamic scalable kd-tree. In: SSTD. pp. 46–65 (2003)
14. Shi, E., Bethencourt, J., Chan, T.H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: SP'07. pp. 350–364 (2007)
15. Singh, C.P.: R-tree implementation of image databases. SIP p. 89 (2011)
16. Tang, F., Zhou, X., Luo, H., Ling, G., Shan, J., Xiao, Y.: Efficient privacy-preserving multi-dimensional range query for cloud-assisted ehealth systems. TSC pp. 2365–2377 (2024)
17. 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: INFOCOM. pp. 2253–2262 (2020)
18. Yang, W., Xu, Y., Nie, Y., Shen, Y., Huang, L.: Trqed: Secure and fast tree-based private range queries over encrypted cloud. In: DASFAA. pp. 130–146 (2018)
19. Zhang, S., Ray, S., Lu, R.: Sorel: Efficient and secure ore-based range query over outsourced data. TBD pp. 1702–1715 (2021)
20. Zhao, C., Song, W., Peng, Z.: How to share medical data belonging to multiple owners in a secure manner. In: APWeb-WAIM. pp. 217–231 (2022)
21. Zheng, Y., Lu, R., Guan, Y., Shao, J., Zhu, H.: Towards practical and privacy-preserving multi-dimensional range query over cloud. TDSC pp. 3478–3493 (2021)