Efficient Privacy-preserving Spatial-keyword Range Query Processing in the Cloud

Mingfeng Jiang

***Abstract*—With the rapid development of location-based services in the mobile Internet, a large amount of spatial-keyword data for range queries is outsourced to the cloud to alleviate the local storage and computational demands. However, directly outsourcing such data to the untrusted cloud could lead to potential privacy issues because of data abuse or breaches. To address this issue, we propose an efficient privacy-preserving spatial-keyword range query processing in the cloud in this paper. First, a novel Gray code-based encoding and vectorization are designed to hide the spatial information of locations and ranges. On the vectors of spatial-keyword data, an equal partition-based keyword-location inverted index (EPKI-index) is constructed. Based on the EPKI-index, a baseline spatial-keyword range query scheme (EPSRQ) is proposed. To improve query efficiency, a novel binary keyword-filtering tree index (BKFtree-index) is designed, and the optimized range query scheme (EPSRQ+) is proposed. In addition, based on the BKFtree-index, EPSRQ+ supports dynamic updates. We formally discussed the security of the whole system, and the experimental results demonstrate that the proposed schemes have better performance than existing works in query efficiency.**

***Index Terms*—cloud computing, privacy-preserving, spatial data, range query**

I. Introduction

With the increasing popularity of location-based services [1]–[6], spatial-keyword queries have gained significant importance with these services. Platforms such as Facebook, Google Maps, etc. enable users to retrieve interested information within a designated search area. With the continuous enrichment of location-based services, there has been an explosive growth in data, thus data owners are inclined to outsource their spatial data to the cloud to reduce storage and information management burdens. However, data outsourcing could cause privacy concerns due to the untrusted cloud [7]. On one hand, the spatial data could disclose sensitive user information, such as location details, textual descriptions, etc. On the other hand, the queries could expose users' preferences and behaviors. Consequently, it is necessary to investigate the privacy-preserving spatial-keyword queries in the cloud.

We mainly focus on the spatial-keyword range query in this paper. As shown in Fig. 1, the figure illustrates a specific area on a map that contains five spatial-keyword items, where (l_i, W_i) is a spatial-keyword item, l_i is a location and W_i is a set of keywords describing this location. For example, $(l_2, \{\text{Mall, Sports}\})$ is a spatial-keyword item. When a user requests Chinese restaurants in the query range r which is drawn by the red rectangle in the figure, he/she performs the spatial-keyword range query $Q = (\{\text{Restaurants, Chinese}\}, r)$ to obtain the query result, and only l_1 meets the query request. Consequently, the query result is l_1.

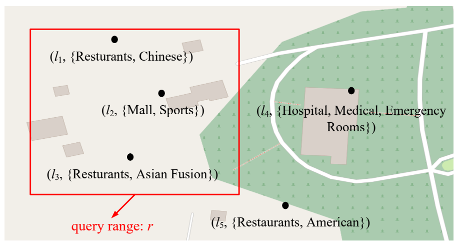

(l_1, {Resturants, Chinese})
(l_2, {Mall, Sports})
(l_3, {Resturants, Asian Fusion})
(l_4, {Hospital, Medical, Emergency Rooms})
(l_5, {Restaurants, American})
query range: r

Fig. 1: An example of range query

To protect the privacy of spatial-keyword data outsourced in the cloud, the intuitive way is to encrypt the data before uploading. However, encrypted data introduces challenges in availability. Therefore, it is a challenge to protect the privacy of outsourced spatial data while maintaining its availability. In recent years, numerous related solutions have been proposed. For example, the literature [8]–[12] realizes the privacy-preserving spatial range queries, but they do not support spatial-keyword queries. The literature [13]–[19] focus on the privacy-preserving spatial-keyword query processing, but these solutions either do not support trapdoor unlinkability preservation or exhibit lower query efficiency.

In this paper, we propose the efficient privacy-preserving spatial-keyword range query processing in the cloud. To protect the privacy of spatial-keyword, the Gray code-based encoding and vectorization methods are designed to encode locations and ranges and then transform the generated codes into vectors. The keyword vectorization is also designed to transform keywords into vectors. Based on the vectors, an equal partition-based keyword-location inverted index (EPKI-index) is constructed and the asymmetric scalar-product-preserving encryption (ASPE) is used to encrypt the index. By using the EPKI index, the baseline spatial-keyword range query scheme (EPSRQ) is proposed. To further improve the query efficiency, an optimized binary keyword-filtering tree index (BKFtree-index) is designed which combines the binary search tree structure with the EPKI-index. Based on the BKFtree-index, the optimized query processing scheme EPSRQ+ is proposed. The game stimulation-based proof is presented to analyze the security of proposed schemes, and the experimental results demonstrate that the proposed schemes have better performance in terms of query efficiency.

The main contributions of this paper are as follows:

- We design a keyword, location and range vectorization method, and then propose an equal partition-based

keyword-location inverted index (EPKI-index). By using the index, we propose the baseline privacy-preserving spatial-keyword range query scheme.
- To improve query efficiency, we propose a novel binary keyword-filtering tree, and then present an optimized privacy-preserving spatial-keyword range query scheme.
- We use the game stimulation-based proof to analyze the security of the proposed schemes and conduct a thorough experimental validation using two real-world datasets.

II. Related Work

In recent years, people have been paying increasing attention to privacy-preserving data queries. Therefore, in this section, we will introduce existing related research, including privacy-preserving spatial query, keyword query and spatial-keyword query.

A. Privacy-preserving Spatial Query

The purpose of spatial queries is to find all the locations in the query range or the k-Nearest-Neighbor (kNN) locations. For example, Zhu et al. [8] proposed an enhanced homomorphic encryption technique over a composite order group. Xu et al. [9] combined the polynomial fitting technique and order-preserving encryption to fit the query range. Lei et al. [10] proposed a secure kNN scheme based on applying projection function composition to test the proximity of two locations. Guo et al. [11] combined Geohash and multi-level indexing structures. Meng et al. [20] encoded locations by prefix encoding and then stored by G-tree. However, these schemes do not support keyword queries.

B. Privacy-preserving Keyword Query

Keyword query involves querying documents in encrypted cloud data based on ranking their textual relevance. For example, Cao et al. [21] introduced the TF-IDF rule and the vector space model in multi-keyword ranked search. Das et al. [22] proposed a tree-based secure ranked search scheme, enabling dynamic operations on the document collection. Zheng et al. [23] constructed a radix tree to store the keyword set. Miao et al [24] proposed a machine learning-based query using k-means clustering algorithm and the balanced binary tree. Zhou et al. [25] proposed the Latent Dirichlet Allocation topic model (LDA) to generate the semantic information. However, these methods cannot be used for queries on spatial data.

C. Privacy-preserving Spatial-keyword Query

Spatial-keyword Query means that the query can be realized while the location is in the query range and contains the query keywords. For example, Cui et al. [13] combined the spatial-text Bloom filter encoding structure and an encrypted R-tree index. Cui et al. [15] constructed a binary tree by the Hilbert curve, where Bloom filters store the index values. Song et al. [16] and Yang et al. [17] employed polynomial fitting technology and ASPE to accomplish secure range queries. Miao et al. [19] improved their scheme by constructing a GR-tree based on Geohash. However, although these schemes support querying both keywords and spatial data, [13], [15] do not support the unlinkability of the trapdoor, while [16], [17], [19] are less efficient in querying despite supporting the unlinkability of the trapdoor.

We propose an efficient privacy-preserving spatial-keyword range query scheme to support the querying of keywords and spatial data while achieving trapdoor unlinkability and efficient query processing.

III. Preliminaries

A. Gray Code Encoding on Spatial Grids

Gray code encoding on spatial grids is a useful geocoding method proposed in [14]. The Gray code is a binary encoding method that any two adjacent codes differ by only one bit. In a T-bit Gray code system, there are 2^T codes, the set of which is denoted as G_T. Assuming that the spatial space is divided into $2^T \times 2^T$ equal grids, the (i, j)-grid is encoded as a $2T$-bit Gray code $g_{i,j} = G_T[i]||G_T[j]$. It is noticeable that the larger T is, the greater the split granularity is.

For example, Fig. 2 shows that a spatial space is divided into $2^2 \times 2^2$ grids. Each grid is encoded as a 4-bit Gray code, such as $g_{1,1} = 0000$, $g_{2,3} = 0111$, etc. The whole Gray codes of 16 grids are shown in Fig. 2.

	00	01	11	10
10	1000 ($g_{4,1}$)	1001 ($g_{4,2}$)	1011 ($g_{4,3}$)	1010 ($g_{4,4}$)
11	1100 ($g_{3,1}$)	1101 ($g_{3,2}$)	1111 ($g_{3,3}$)	1110 ($g_{3,4}$)
01	0100 ($g_{2,1}$)	0101 ($g_{2,2}$)	0111 ($g_{2,3}$)	0110 ($g_{2,4}$)
00	0000 ($g_{1,1}$)	0001 ($g_{1,2}$)	0011 ($g_{1,3}$)	0010 ($g_{1,4}$)

Fig. 2: An Example of Gray codes on $2^2 \times 2^2$ grids

B. Asymmetric Scalar-Product-Preserving Encryption

Asymmetric Scalar-Product-Preserving Encryption (ASPE) [11] is an asymmetric encryption scheme that supports preserving scalar product operations, which is used to compute the scalar product of two vectors in the ciphertext without revealing the plaintext. Assuming that p and q are two m-dimensional plaintext vectors, M is a random $m \times m$-dimensional invertible matrice which is used as the secret key, and $Enc_M(\cdot)$ is the vector encryption function. p and q are encrypted into $\widetilde{p} = Enc_M(p) = M^T p$ and $\widetilde{q} = Enc_M(p) = M^{-1}q$, respectively. Because of $\widetilde{p} \cdot \widetilde{q} = (M^T p)^T M^{-1} q = p^T M M^{-1} q = p \cdot q$, we have that the scalar product result of the encrypted vectors $\widetilde{p}$ and $\widetilde{q}$ equals to that of the plaintext vectors p and q. All vectors in the proposed schemes are encrypted based on ASPE.

IV. Problem Formulation

A. System Model and Threat Model

System model. In this paper, the system model consists of three components: Data Owner (DO), Cloud Server (CS), and Data User (DU), which is the same as the models in [11], [15], [18]–[20], [24]. DO is responsible for generating indexes and encrypting data, and then uploading them to CS. DO also needs to share the secret key and other relevant information with DU. DU generates query trapdoors and submits these trapdoors to CS. CS performs the query algorithm on the encrypted index and returns the encrypted query result to DU.

Threat model. Similar to references [11], [15], [18]–[20], [24], this paper adopts the "honest but curious" threat model. In this model, CS provides query services for DU, but it may also be curious and attempt to capture or analyze the stored data or query information. Assuming that DU is trustworthy and the transmission channel is secure, CS should not be able to obtain true plaintext information from the outsourced data.

B. Framework

To introduce the proposed schemes concisely, we present the framework which consists of two modules, the setup module and the query module. The setup module is used to pre-process and encrypt the dataset, while the query module is used to accomplish the range query. These modules are built upon four essential algorithms that form the foundation of the schemes:

- ***GenKey***: This algorithm is used to generate secret keys.
- ***GenIndex***: This algorithm is used to generate the encrypted index for the dataset.
- ***GenTrapdoor***: This algorithm is used to convert the query request into a trapdoor.
- ***Query***: This algorithm is used to accomplish the range query and return the query result.

C. Security Definition

We utilize a game simulation [26] to define the security of our schemes. For simplicity, let Π denote the privacy-preserving spatial-keyword range query scheme, and let $\mathcal{A}$ and $\mathcal{B}$ represent the adversary and the simulator, respectively. The scheme is considered perfect $\mathcal{F}$-adaptively secure if the ASPE-based vector encryption scheme can achieve unlinkability. The security definition is as follows:

Definition 1. Perfect Indistinguishability. To prove the perfect indistinguishability of the vector encryption scheme, we design the indistinguishable experiment $exp_{\mathcal{A}}^{Ind}(\lambda)$. Given an adversary $\mathcal{A}$ the adaptive access to the encryption oracle and decryption oracle, and the number of query times is less than m–2, where m is the length of the vectors to be encrypted. The details of $exp_{\mathcal{A}}^{Ind}(\lambda)$ are shown as follow. The vector encryption scheme can achieve perfect indistinguishability if $|Pr[exp_{\mathcal{A}}^{Ind}(\lambda) = 1] - \frac{1}{2}| = 0$ holds for any adversary $\mathcal{A}$.

```
Indistinguishable experiment: $exp_{\mathcal{A}}^{Ind}(\lambda)$
1: $k \leftarrow GenKey(1^{\lambda})$
2: $\{v_{c_0}, v_{c_1}\} \leftarrow \mathcal{A}$
3: $b \xleftarrow{\$} \{0,1\}$
4: $\widetilde{v}_{c_b} \leftarrow Enc_k(v_{c_b})$
5: $b' \leftarrow \mathcal{A}(\widetilde{v}_{c_b}, v_{c_0}, v_{c_1})$
```

Definition 2. Unlinkability. The vector encryption scheme achieves unlinkability if it satisfies the perfect indistinguishability in Definition 1.

Definition 3. Leakage Functions. Against the proposed scheme Π, $\mathcal{F}_{\Pi} = \{\mathcal{F}^{GK}, \mathcal{F}^{GI}, \mathcal{F}^{GT}, \mathcal{F}^{Q}\}$ is a set of leakage functions corresponding to the algorithms *GenKey*, *GenIndex*, *GenTrapdoor* and *Query*, respectively. Specifically, the leakages are described as follows:

- $Leak_{SP}$ is the search pattern leakage, which reflects the encrypted results acquired by the dataset and trapdoors.
- $Leak_{AP}$ is the access pattern leakage, which is the identities of the returned encrypted spatial-keyword items.

In $\mathcal{F}_{\Pi}$, $\mathcal{F}^{GK}$, $\mathcal{F}^{GI}$ and $\mathcal{F}^{GT}$ correspond to $Leak_{SP}$, and $\mathcal{F}^{Q}$ corresponds to $Leak_{AP}$. The query results are ciphertext generated by symmetric encryption, thus $Leak_{AP}$ will not leak the plaintext information of results. The security of $Leak_{SP}$ will be proved in the security analysis section (Section VII).

Definition 4. Perfect $\mathcal{F}$-adaptively secure. Given a scheme Π, for an adversary $\mathcal{A}$, The advantage for $\mathcal{A}$ to win the guess in $exp_{\mathcal{A}}^{Ind}(\lambda)$ is denoted as $Adv_{\mathcal{A}}(k)$. If $Adv_{\mathcal{A}}(k) = |Pr[b' = b] - \frac{1}{2}| = 0$ holds, then Π can achieve perfect indistinguishability and further Π can achieve unlinkability. Thus Π is perfect $\mathcal{F}$-adaptively secure.

D. Problem Description

Definition 5. Spatial-keyword Dataset. The spatial-keyword dataset D is a set of n spatial-keyword items, $D = \{(l_1, W_1), (l_2, W_2), \cdots, (l_n, W_n)\}$, where $(l_i, W_i) \in D$ is a spatial-keyword item and W_i is a set of keywords describing the 2-dimensional location point l_i.

Given a spatial-keyword dataset D, a spatial-keyword range query $Q = \{W_Q, R_Q\}$ is to retrieve the query result R consisting of a subset of spatial-keyword items in D, the keyword and location of each item in which belong to query keywords W_Q and the query range R_Q, respectively, i.e.,

$$R = \{(l_t, W_t) | (l_t, W_t) \in D \wedge W_t \subseteq W_Q \wedge l_t \prec R_Q\} \quad (1)$$

where $l_t \prec R_Q$ represents that the location l_t is in R_Q.

The privacy-preserving spatial-keyword range query processing should satisfy two goals:

Query Efficiency. The proposed scheme is designed to achieve efficient spatial-keyword range queries.

Privacy Preservation. The proposed scheme is designed to protect the privacy of various aspects, including spatial-keyword privacy, query privacy and trapdoor unlinkability. Here, the trapdoor unlinkability represents that CS is prevented from determining whether two distinct trapdoors are generated from the same query.

V. The Baseline Query Processing Scheme

In this section, we first present the vectorizations of keywords, locations and ranges. Then, we propose an equal partition-based keyword-location inverted index (EPKI-index) storing vectors of spatial-keyword data. At last, we propose the details of the baseline privacy-preserving spatial-keyword query processing scheme (EPSRQ).

A. Keyword, Location and Range Vectorization

Definition 6. Keyword Dictionary. The keyword dictionary, denoted as $W = \{w_1, w_2, \cdots, w_m\}$, is a keyword set composed by the union of the keywords in every spatial-keyword item of D, i.e.,

$$W = \bigcup_{(l_i, W_i) \in D} W_i, \tag{2}$$

where $w_j \in W$ is a keyword.

Definition 7. Keyword Vector. Given a keyword $w_i \in W$, the keyword vector of w_i is a m-dimensional vector, denoted as V_{w_i}, and the j-th dimension $V_{w_i}[j]$ is set follows Eq. 3.

$$V_{w_i}[j] = \begin{cases} 1 & j = i \\ 0 & otherwise \end{cases} \tag{3}$$

For a keyword set $W_Q \subset W$, the keyword vector of W_Q is the result of performing the bit-wise OR operation on the keyword vectors of all keywords in W_Q, i.e.,

$$V_{W_Q} = \bigvee_{w_i \in W_Q} V_{w_i}, \tag{4}$$

where $\bigvee$ is the OR operator.

Lemma 1. Given a keyword w_i, a keyword set W_Q, and the keyword vectors of them are V_{w_i} and V_{W_Q}, we have

$$w_i \in W_Q \iff V_{w_i} \cdot V_{W_Q} \neq 0. \tag{5}$$

Proof. According to Definition 7, V_{w_i} and V_{W_Q} are both m-dimensional vectors, thus we have Eq. 6.

$$V_{w_i} \cdot V_{W_Q} = \sum_{j=1}^{m} V_{w_i}[j] \cdot V_{W_Q}[j] \tag{6}$$

We give the proof from sufficiency and necessity of this lemma.

- *Sufficiency.* If $w_i \in W_Q$ holds, according to Eq. 3 and Eq. 4, we can deduce that $V_{w_i}[i] = V_{W_Q}[i] = 1$. According to Eq. 6, we have $V_{w_i}[i] = V_{W_Q}[i] = 1$, thus $V_{w_i} \cdot V_{W_Q} \neq 0$ holds.
- *Necessity.* If $V_{w_i} \cdot V_{W_Q} \neq 0$ holds, according to Eq. 6, we deduce that $\exists V_{w_i}[j] \cdot V_{W_Q}[j] = 1$. According to Eq. 4, since $\exists V_{w_i}[j] \cdot V_{W_Q}[j] = 1$, we have $\exists V_{w_i}[j] = V_{W_Q}[j] = 1$, thus $w_i \in W_Q$ holds.

According to the proof of sufficiency and necessity, we deduce that Lemma 1 holds. ■

To introduce location encoding and vectorization, we assume that the spatial space is equally divided into $2^T \times 2^T$ grids. According to the Gray code encoding on spatial grids, the Gray code of (i, j)-grid is $g_{i,j} = G_T[i]||G_T[j]$, where (i, j)-grid is the grid located in the i-th row and j-th column, and G_T is the T-bit Gray code set. Thus, the Gray code $g_{i,j}$ is a $2T$-bit string.

Definition 8. Gray Code-based Location Encoding. Given a location l_u which is assumed in the (i, j)-grid and the $2T$-bit Gray code of the grid is $g_{i,j}$, the location code of l_u is a $4T$-bit string denoted as C_{l_u} and generated follows Eq. 7,

$$C_{l_u}[2k-1, 2k] = \begin{cases} 01 & g_{i,j}[k] = 0 \\ 10 & g_{i,j}[k] = 1 \end{cases}, k \in \{1, 2, \cdots, 2T\} \tag{7}$$

where $C_{l_u}[2k-1, 2k]$ is the 2-bit string composed by the $(2k-1)$-th and $2k$-th bits of C_{l_u}, and $g_{i,j}[k]$ is the k-th bit of $g_{i,j}$.

Definition 9. Location Vector. Given a location l_u whose location code is a $4T$-bit string C_{l_u}, the location vector of l_u is a $(4T+1)$-dimensional vector denoted as V_{l_u} and generated follows Eq. 8.

$$V_{l_u}[i] = \begin{cases} C_{l_u}[i] & i \in \{1, 2, \cdots, 4T\} \\ 2T & i = 4T + 1 \end{cases} \tag{8}$$

Definition 10. Grid-based Minimum Bounding Rectangle (GMBR). Give a range r (such as a rectangle, a circle, etc), the GMBR of r, denoted as $\mathcal{R}$, is a rectangle composed of the minimum girds that can cover r. The set of Gray codes of the grids in $\mathcal{R}$ is denoted as $Gray(\mathcal{R})$.

Definition 11. Gray Code-based Grid Re-encoding. Given a (i, j)-grid whose Gray code is $g_{i,j}$, the grid code of (i, j)-grid is a $4T$-bit string denoted as $E_{i,j}$ and generated by re-encoding the Gray codes of grids. The re-encoding rule follows Eq. 9,

$$E_{i,j}[2k-1, 2k] = \begin{cases} 01 & g_{i,j}[k] = 0 \\ 10 & g_{i,j}[k] = 1 \end{cases}, k \in \{1, 2, \cdots, 2T\} \tag{9}$$

where $E_{i,j}[2k–1, 2k]$ is the 2-bit string composed by the $(2k–1)$-th and $2k$-th bits of $E_{i,j}$, and $g_{i,j}[k]$ is the k-th bit of $g_{i,j}$.

Definition 12. Gray Code-based Range Encoding. Given a range r and the GMBR of r is $\mathcal{R}$, the range code of r is a $4T$-bit string denoted as C_r and generated follows Eq. 10,

$$C_r[2k-1, 2k] = \bigvee_{g_{i,j} \in Gray(\mathcal{R})} E_{i,j}[2k-1, 2k], k \in \{1, 2, \cdots, 2T\} \tag{10}$$

where $E_{i,j}$ is the grid code of (i, j)-grid in $\mathcal{R}$.

Definition 13. Range Vector. iven a range r and the range code of r is C_r, the range vector of r is a $(4T+1)$-dimensional vector, which is denoted as V_r and generated follows Eq. 11.

$$V_r[i] = \begin{cases} C_r[i] & i \in \{1, 2, \cdots, 4T\} \\ -1 & i = 4T + 1 \end{cases} \tag{11}$$

Lemma 2. Given a location l_u and a (i, j)-grid, the location and grid codes are C_{l_u} and $E_{i,j}$, respectively, if l_u is in the (i, j)-grid, then we have $C_{l_u} = E_{i,j}$.

Proof. Because the location l_u is in the (i, j)-grid, according to Definition 8 and Definition 11, both C_{l_u} and $E_{i,j}$ are generated by the Gray code of the grid $g_{i,j}$. Since the code generation rules of C_{l_u} and $E_{i,j}$ are the same as shown in Eq. 7 and Eq. 9, we can deduce $C_{l_u} = E_{i,j}$. As a result, Lemma 2 holds. ■

Lemma 3. Given a location l_u and a range r, the location and range vectors of which are V_{l_u} and V_r, respectively, we have

$$l_u \prec r \iff V_{l_u} \cdot V_r = 0, \tag{12}$$

where $\prec$ represents that the location l_u is in the range r.

Proof. We assume that the location l_u is in the (i,j)-grid, the GMBR of r is $\mathcal{R}$, the location code of l_u is C_{l_u} and the range code of r is C_r. According to Definition 9 and 13, V_{l_u} and V_r are both $(4T+1)$-dimensional vectors, thus we have

$$\begin{aligned} V_{l_u} \cdot V_r &= V_{l_u}[1,2,\cdots,4T] \cdot V_r[1,2,\cdots,4T] \\ &\quad + V_{l_u}[4T+1] \cdot V_r[4T+1] \\ &= \sum_{k=1}^{2T} V_{l_u}[2k-1,2k] \cdot V_r[2k-1,2k] - 2T. \end{aligned} \tag{13}$$

We give the proof from sufficiency and necessity of this lemma.

- *Sufficiency.* If $l_u \prec r$ holds, then the (i,j)-grid where l_u located is in $\mathcal{R}$. According to Definition 12, C_r is generated by the Gray code of the grids in $\mathcal{R}$, thus we deduce that for each $k \in \{1,2,\cdots,2T\}$, $C_r[2k-1,2k] = E_{i,j}[2k-1,2k]$ or $C_r[2k-1,2k] = 11$ holds. Lemma 2 indicates $C_{l_u} = E_{i,j}$, thus we have $C_r[2k-1,2k] = C_{l_u}[2k-1,2k]$ or $C_r[2k-1,2k] = 11$, which means $C_{l_u}[2k-1,2k] \cdot C_r[2k-1,2k] = 1$. According to Eq. 8 and Eq. 11, we have $V_{l_u}[2k-1,2k] \cdot V_r[2k-1,2k] = 1$, thus $\sum_{k=1}^{2T} V_{l_u}[2k-1,2k] \cdot V_r[2k-1,2k] = 2T$ holds. According to Eq. 13, we can deduce $V_{l_u} \cdot V_r = 0$.
- *Necessity.* If $V_{l_u} \cdot V_r = 0$ holds, according to Eq. 13, we have $\sum_{k=1}^{2T} V_{l_u}[2k-1,2k] \cdot V_r[2k-1,2k] = 2T$. Since $V_{l_u}[2k-1,2k] \in \{01,10\}$ and $V_r[2k-1,2k] \in \{01,10,11\}$, we can deduce $V_{l_u}[2k-1,2k] \cdot V_r[2k-1,2k] = 1$, thus $V_{l_u}[2k-1,2k] = V_r[2k-1,2k]$ or $V_r[2k-1,2k] = 11$ holds, and then $C_{l_u}[2k-1,2k] = C_r[2k-1,2k]$ or $C_r[2k-1,2k] = 11$ is deduced. According to Definition 12, C_r is generated by $E_{i,j}$ which equals to C_{l_u} according to Lemma 2, we have that the (i,j)-grid is in the range $\mathcal{R}$. Since l_u is in the (i,j)-grid, we deduce that $l_u \prec r$ holds.

According to the proof of sufficiency and necessity, we deduce that Lemma 3 holds. ■

Lemma 3 indicates that the scalar production result between a location vector and a range vector represents whether a location is in a range. This feature can be used to implement privacy-preserving range queries in locations.

To clearly illustrate the above definitions and lemmas, we give an example as shown in Fig. 3. There are 5 locations $\{l_1,l_2,l_3,l_4,l_5\}$ and a range r. Taking l_4 as an example, the location vector is $V_{l_4} = (0,1,1,0,1,0,1,0,4)$. The GMBR of r is $Gray(\mathcal{R}) = \{0001,0011,0101,0111\}$ and the range vector is $V_r = (0,1,1,1,1,1,1,0,-1)$. Thus, we have $V_{l_4} \cdot V_r = 0$, and l_4 is in the range r, thus Lemma 3 is verified.

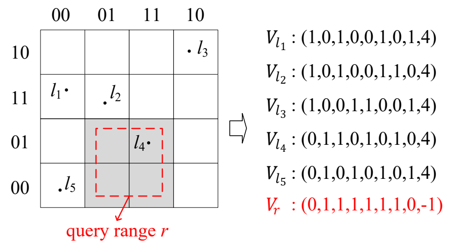

00
01
11
10
10
11
01
00
l_1
l_2
l_3
l_4
l_5
query range r
V_{l_1} : (1,0,1,0,0,1,0,1,4)
V_{l_2} : (1,0,1,0,0,1,1,0,4)
V_{l_3} : (1,0,0,1,1,0,0,1,4)
V_{l_4} : (0,1,1,0,1,0,1,0,4)
V_{l_5} : (0,1,0,1,0,1,0,1,4)
V_r : (0,1,1,1,1,1,1,0,-1)

Fig. 3: An example of location vectors and range vector

B. Equal Partition-based Keyword-location Inverted Index

Definition 14. Location Vector Set of Keyword. Given a keyword $w_i \in W$, the location vector set of w_i consists of the location vectors of locations corresponding to w_i, which is denoted as $LVS(w_i)$,

$$LVS(w_i) = \{V_{l_t} | (l_t, W_t) \in D \wedge w_i \in W_t\} \tag{14}$$

where V_{l_t} is the location vector of the location l_t and W_t is the corresponding keywords of l_t.

Definition 15. Equal Partition-based Keyword-location Inverted Index (EPKI-index). Given a partition threshold γ, the EPKI-index of D is a set of two-element tuples and each tuple consists of a keyword vector and a corresponding posting list having γ location vectors. The EPKI-index is denoted as KL,

$$KL = \bigcup_{w_i \in W} KL_i \tag{15}$$

where KL_i is a set of two-element tuple corresponding to w_i,

$$KL_i = \{(V_{w_i}, PL_{i,j}) | j \in \{1,2,\cdots,\tau_i\}\}, \tag{16}$$

V_{w_i} is the keyword vector of the keyword w_i, $PL_{i,j}$ is the posting list having γ location vectors corresponding to w_i, and $\tau_i = \lceil \frac{|LVS(w_i)|}{\gamma} \rceil$. Each location vector in $PL_{i,j}$ corresponds to a spatial-keyword item. For a keyword w_i, KL_i is generated according to the following steps:

1) Divide the location vector set $LVS(w_i)$ into equal-length posting lists $\{PL_{i,1}, PL_{i,2}, \cdots, PL_{i,\tau_i}\}$, where $|PL_{i,1}| = |PL_{i,2}| = \cdots = |PL_{i,\tau_i-1}| = \gamma$ and $|PL_{i,\tau_i}| \le \gamma$.
2) If $|PL_{i,\tau_i}| < \gamma$ holds, then create $\gamma - |PL_{i,\tau_i}|$ phantom location vectors according to Eq. 17,

$$\begin{cases} V_{ph}[i] = C_{ph}[i] & i \in \{1,2,\cdots,4T\} \\ V_{ph}[i] \neq 2T & i = 4T+1 \end{cases}, \tag{17}$$

where V_{ph} is a phantom location vector and C_{ph} is a $4T$-bit string. The construction of C_{ph} follows Eq. 18,

$$C_{ph}[2k-1,2k] = rand\{01,10\} \; k \in \{1,2,\cdots,2T\}, \tag{18}$$

where $rand\{01,10\}$ returns a randomly chosen two-bit string from $\{01,10\}$.
3) Use the keyword vector V_{w_i} and the generated equal-length posting lists $\{PL_{i,1}, PL_{i,2}, \cdots, PL_{i,\tau_i}\}$ to construct the two-element tuple set $KL_i = \{(V_{w_i}, PL_{i,j}) | j \in \{1,2,\cdots,\tau_i\}\}$.

According to Definition 15, we give the construction algorithm of the EPKI-index which is shown in Algorithm 1. We give an example to describe the EPKI-index as shown in

Fig. 4. Assuming that there are nine locations $\{l_1, l_2, \cdots, l_9\}$ corresponding to four keywords $\{w_1, w_2, w_3, w_4\}$ and each location corresponds to a keyword, the EPKI-index with the partition threshold $\gamma = 2$ is presented in the right area of Fig. 4.

Algorithm 1: *GenEPKI*(W, D, γ)

```
   Input: the keyword dictionary W, the spatial-keyword
          dataset D, and the partition threshold $\gamma$
   Output: the EPKI-index $KL$
1  Initialize $KL = \varnothing$;
2  for each $w_i \in W$ do
3      if $LVS(w_i) \leq \gamma$ then
4          Create an empty posting list $PL_{i,1}$, and add the
           location vectors of $LVS(w_i)$ into $PL_{i,1}$;
5          Create $\gamma - |PL_{i,1}|$ phantom location vectors
           and add them into $PL_{i,1}$;
6          $KL_i = \{(V_{w_i}, PL_{i,1})\}$;
7      else
8          Divide $LVS(w_i)$ into $\tau_i = \lceil \frac{|LVS(w_i)|}{\gamma} \rceil$ posting
           lists $\{PL_{i,1}, \cdots, PL_{i,\tau_i}\}$, where $|PL_{i,1}| = \cdots$
           $= |PL_{i,\tau_i - 1}| = \gamma$ and $|PL_{i,\tau_i}| \leq \gamma$;
9          Create $\gamma - |PL_{i,\tau_i}|$ phantom location vectors,
           and add them into $PL_{i,\tau_i}$;
10         $KL_i = \{(V_{w_i}, PL_{i,j}) | j \in \{1, 2, \cdots, \tau_i\}\}$;
11     Append all the items of $KL_i$ to $KL$;
12 return $KL$;
```

In the EPKI-index, locations corresponding to the same keyword are aggregated in a few posting lists, and hence they can be quickly obtained when the keyword is queried. Moreover, because the posting lists in the index are equal in length, the information about hot or cold keywords deduced according to the length of posting lists in the traditional inverted index is constrained, which is helpful for privacy preservation.

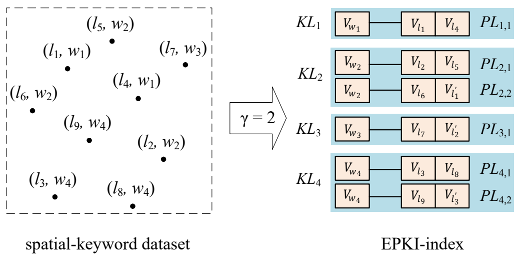

(l_5, w_2)
(l_1, w_1)
(l_7, w_3)
(l_4, w_1)
(l_6, w_2)
(l_9, w_4)
(l_2, w_2)
(l_3, w_4)
(l_8, w_4)
$\gamma = 2$
KL_1
V_{w_1} V_{l_1} V_{l_4} $PL_{1,1}$
KL_2
V_{w_2} V_{l_2} V_{l_5} $PL_{2,1}$
V_{w_2} V_{l_6} $V_{l'_1}$ $PL_{2,2}$
KL_3
V_{w_3} V_{l_7} $V_{l'_2}$ $PL_{3,1}$
KL_4
V_{w_4} V_{l_3} V_{l_8} $PL_{4,1}$
V_{w_4} V_{l_9} $V_{l'_3}$ $PL_{4,2}$
spatial-keyword dataset
EPKI-index

Fig. 4: An example of EPKI-index ($\gamma = 2$)

C. The Baseline Query Processing Scheme

In this section, we give the details of the baseline efficient privacy-preserving spatial-keyword query processing scheme (EPSRQ), which has two modules, the setup module and the query processing module.

• ***Algorithms in Setup Module***

$\boldsymbol{sk} \leftarrow$ ***GenKey*** $(\mathbf{1}^{\boldsymbol{\lambda}})$. Given a security parameter λ, DO generates a set of secret keys $sk = \{k, s_1, M_{11}, M_{12}, s_2, M_{21}, M_{22}\}$ and shares sk with DU. In sk, k is a symmetric encryption key which is used to encrypt the spatial-keyword dataset; s_1 and s_2 are random m-dimensional and $(4T+1)$-dimensional bit vectors; M_{11} and M_{12} are $m \times m$-dimensional random invertible matrices; M_{21} and M_{22} are $(4T+1) \times (4T+1)$-dimensional random invertible matrices. $\{s_1, M_{11}, M_{12}\}$ is used to encrypt keyword vectors while $\{s_2, M_{21}, M_{22}\}$ is used to encrypt location vectors and range vectors.

$\{\widetilde{\boldsymbol{KL}}, \widetilde{\boldsymbol{D}}\} \leftarrow$ ***GenIndex***$(\boldsymbol{W}, \boldsymbol{D}, \boldsymbol{sk}, \boldsymbol{\gamma})$. DO uses the algorithm *GenIndex* to generate the encrypted EPKI-index by inputting the keyword dictionary W, the spatial-keyword dataset D, the partition threshold γ and the secret key set sk. The details of *GenIndex* are described as the following steps:

1) DO uses the key $k \in sk$ to encrypt each spatial-keyword item $(l_t, W_t) \in D$ into the ciphertext $(\widetilde{l_t}, \widetilde{W_t})$, and the encrypted spatial-keyword dataset $\widetilde{D}$ is generated.
2) DO uses the *GenEPKI* algorithm (Algorithm 1) to generate the plaintext EPKI-index KL.
3) For each $(V_{w_i}, PL_{i,j}) \in KL$ and each $V_{l_t} \in PL_{i,j}$, DO uses the *EncVec* algorithm (Algorithm 2) to encrypt V_{w_i} and V_{l_t} into $\widetilde{V}_{w_i}$ and $\widetilde{V}_{l_t}$, respectively, according to Eq. 19 and Eq. 20. The encrypted location vector $\widetilde{V_{l_t}}$ corresponds to the encrypted spatial-keyword item $(\widetilde{l_t}, \widetilde{W_t}) \in \widetilde{D}$.

$$\begin{aligned} \widetilde{V}_{w_i} &= \mathit{EncVec}(V_{w_i}, s_1, M_{11}, M_{12}, true) \\ &= \{M_{11}^T V'_{w_i}, M_{12}^T V''_{w_i}\} \end{aligned} \tag{19}$$

$$\begin{aligned} \widetilde{V}_{l_t} &= \mathit{EncVec}(V_{l_t}, s_2, M_{21}, M_{22}, true) \\ &= \{M_{21}^T V'_{l_t}, M_{22}^T V''_{l_t}\} \end{aligned} \tag{20}$$

Afterwards, the encrypted EPKI-index $\widetilde{KL}$ is generated.

After the EPKI-index and spatial-keyword dataset are encrypted, DO outsources the encrypted data to CS and the setup module is accomplished.

Algorithm 2: *EncVec*$(V, s, M_1, M_2, flag)$

```
  Input: the vector $V$, the random bit vector $s$, and two
         random invertible matrices $M_1$ and $M_2$
  Output: the encrypted vector $\widetilde{V}$
1 if $flag = true$ then
2     Split $V$ into $\{V', V''\}$, according to the rules:
      $\begin{cases} V'[i] = V''[i] = V[i], & s[i] = 0 \\ V'[i] + V''[i] = V[i], & s[i] = 1 \end{cases}$;
3     $\widetilde{V} = \{M_1^T V', M_2^T V''\}$;
4 else
5     Split $V$ into $\{V', V''\}$, according to the rules:
      $\begin{cases} V'[i] + V''[i] = V[i], & s[i] = 0 \\ V'[i] = V''[i] = V[i], & s[i] = 1 \end{cases}$;
6     $\widetilde{V} = \{M_1^{-1} V', M_2^{-1} V''\}$;
7 return $\widetilde{V}$;
```

• ***Algorithms in Query Module***

TD $\leftarrow$ ***GenTrapdoor***$(\boldsymbol{Q}, \boldsymbol{sk})$. Given a query $Q = \{W_Q, R_Q\}$, where W_Q is the query keyword set and R_Q is the query range, DU generates the trapdoor $TD = \{\widetilde{V}_{W_Q}, \widetilde{V}_{\mathcal{R}}\}$ where $\widetilde{V}_{W_Q}$ and $\widetilde{V}_{\mathcal{R}}$ are the encrypted query keyword vector and the encrypted query range vector, respectively. Then, DU transmits TD as the query command to CS. The generation of the trapdoor is described as follows:

1) For the query keyword set $W_Q \in Q$, DU vectorizes it as V_{W_Q} according to Definition 7, and then encrypts V_{W_Q} into $\widetilde{V}_{W_Q}$ by using the *EncVec* algorithm, where

$$\begin{aligned}\widetilde{V}_{W_Q} &= \mathit{EncVec}(V_{W_Q}, s_1, M_{11}, M_{12}, false) \\ &= \{M_{11}^{-1}V'_{W_Q}, M_{12}^{-1}V''_{W_Q}\}.\end{aligned} \tag{21}$$

2) For the query range $R_Q \in Q$, DU generates the corresponding GMBR $\mathcal{R}$ according to Definition 10. Afterwards, DU vectorizes $\mathcal{R}$ as the range vector $V_{\mathcal{R}}$ according to Definition 12, and then encrypts $V_{\mathcal{R}}$ into $\widetilde{V}_{\mathcal{R}}$ by using the *EncVec* algorithm, where

$$\begin{aligned}\widetilde{V}_{\mathcal{R}} &= \mathit{EncVec}(V_{\mathcal{R}}, s_2, M_{21}, M_{22}, false) \\ &= \{M_{21}^{-1}V'_{\mathcal{R}}, M_{22}^{-1}V''_{\mathcal{R}}\}.\end{aligned} \tag{22}$$

3) After generating $\widetilde{V}_{W_Q}$ and $\widetilde{V}_{\mathcal{R}}$, the trapdoor $TD = \{\widetilde{V}_{W_Q}, \widetilde{V}_{\mathcal{R}}\}$ is constructed.

According to the above trapdoor generation procedures and the *EncVec* algorithm, even for the same spatial-keyword range queries running in different times, the generated trapdoors are totally different, which indicates that CS cannot identify the same queries and thus the trapdoor unlinkability is preserved.

$\boldsymbol{R} \leftarrow$ ***Query***$(\widetilde{\boldsymbol{D}}, \widetilde{\boldsymbol{KL}}, \boldsymbol{TD})$. Once receiving a query trapdoor $TD = \{\widetilde{V}_{W_Q}, \widetilde{V}_{\mathcal{R}}\}$ from DU, CS runs Algorithm 3 to perform the privacy-preserving spatial-keyword range query processing. When the query result R is determined, CS returns it to DU. The details of the query algorithm are presented in Algorithm 3.

Algorithm 3: *Query*$(\widetilde{D}, \widetilde{KL}, TD)$

```
Input: the encrypted spatial-keyword dataset $\widetilde{D}$, the
       encrypted EPKI-index $\widetilde{KL}$, and the trapdoor
       $TD = \{\widetilde{V}_{W_Q}, \widetilde{V}_{\mathcal{R}}\}$
Output: the query result $R$
1 for each $(\widetilde{V}_{w_i}, \widetilde{PL}_{i,j}) \in \widetilde{KL}$ do
2     Initialize a candidate result set $CR_i$ for $w_i$;
3     if $\widetilde{V}_{w_i} \cdot \widetilde{V}_{W_Q} \neq 0$ then
4         for each $\widetilde{V}_{l_t} \in \widetilde{PL}_{i,j}$ do
5             if $\widetilde{V}_{l_t} \cdot \widetilde{V}_{\mathcal{R}} = 0$ then
6                 Add the encrypted item $(\widetilde{l_t}, \widetilde{W_t})$
                   corresponding to $\widetilde{V}_{l_t}$ into $CR_i$;
7 $R = \bigcap_{w_i \in W_Q} CR_i$;
8 return $R$;
```

In Algorithm 3, each encrypted tuple $(\widetilde{V}_{w_i}, \widetilde{PL}_{i,j})$ in the encrypted index $\widetilde{KL}$ is tranversed to filter those qualified tuples by checking whether $\widetilde{V}_{w_i} \cdot \widetilde{V}_{W_Q} \neq 0$ holds. Here, $\widetilde{V}_{w_i} \cdot \widetilde{V}_{W_Q} \neq 0$ indicates $V_{w_i} \cdot V_{W_Q} \neq 0$ according to Eq. 23, and thus we have $w_i \in W_Q$ according to Lemma 1. It means that the spatial-keyword items corresponding to the locations stored in $\widetilde{PL}_{i,j}$ could satisfy the query request.

$$\begin{aligned}&\widetilde{V}_{w_i} \cdot \widetilde{V}_{W_Q} \\ &= \{M_{11}^T V'_{w_i}, M_{12}^T V''_{w_i}\} \cdot \{M_{11}^{-1}V'_{W_Q}, M_{12}^{-1}V''_{W_Q}\} \\ &= (M_{11}^T V'_{w_i})^T (M_{11}^{-1}V'_{W_Q}) + (M_{12}^T V''_{w_i})^T (M_{12}^{-1}V''_{W_Q}) \\ &= V'_{w_i} \cdot V'_{W_Q} + V''_{w_i} \cdot V''_{W_Q} \\ &= V_{w_i} \cdot V_{W_Q}\end{aligned} \tag{23}$$

Afterwards, each encrypted location vector $\widetilde{V}_{l_t} \in \widetilde{PL}_{i,j}$ is checked whether $\widetilde{V}_{l_t} \cdot \widetilde{V}_{\mathcal{R}} = 0$ holds to find the locations in the query range. Here, $\widetilde{V}_{l_t} \cdot \widetilde{V}_{\mathcal{R}} = 0$ indicates $V_{l_t} \cdot V_{\mathcal{R}} = 0$ according to Eq. 24, thus we have that the location l_t is in the query range R_Q according to Lemma 3. The corresponding encrypted spatial-keyword item $(\widetilde{l_t}, \widetilde{W_t})$ is a candidate for the query result and added into the candidate result set CR_i.

$$\begin{aligned}&\widetilde{V}_{l_t} \cdot \widetilde{V}_{\mathcal{R}} \\ &= \{M_{21}^T V'_{l_t}, M_{22}^T V''_{l_t}\} \cdot \{M_{21}^{-1}V'_{\mathcal{R}}, M_{22}^{-1}V''_{\mathcal{R}}\} \\ &= (M_{21}^T V'_{l_t})^T (M_{21}^{-1}V'_{\mathcal{R}}) + (M_{22}^T V''_{l_t})^T (M_{22}^{-1}V''_{\mathcal{R}}) \\ &= V'_{l_t} \cdot V'_{\mathcal{R}} + V''_{l_t} \cdot V''_{\mathcal{R}} \\ &= V_{l_t} \cdot V_{\mathcal{R}}\end{aligned} \tag{24}$$

According to Algorithm 3, for each encrypted spatial-keyword item $(\widetilde{l_t}, \widetilde{W_t})$ in the intersection of the candidate result sets, the location l_t is in the query range and the query keywords belong to the keyword set W_t, which indicates that $(\widetilde{l_t}, \widetilde{W_t})$ belongs to the query result. Therefore, the intersection of the candidate result sets is the final query result.

D. Dynamic Update

Based on the construction of the EPKI-index, EPSRQ supports dynamic updates, including insert and delete.

Insert. When performing an insert operation, regardless of whether the keywords of the items to be inserted are already in the EPKI-index KL or not, we first construct two-element tuples $(V_{w_i}, PL_{i,j})$ for the items to be inserted. Then we insert all the pairs $(V_{w_i}, PL_{i,j})$ into KL according to Algorithm 1, and finally complete the insert operation.

Delete. When performing a delete operation, for each item to be deleted, similar to the query operation, we first locate it in KL according to Algorithm 3 and then replace it with a phantom location vector, which can delete the item without affecting the original structure.

VI. The Optimized Query Processing Scheme

In this section, we propose an optimized privacy-preserving spatial-keyword query processing scheme (EPSRQ+).

A. Binary Keyword-filtering Tree Index

Definition 16. Binary Keyword-filtering Tree Index (BKFtree-index). The binary keyword-filtering tree index is a binary search tree for fast querying for keywords, which is denoted as $\mathcal{L}$. Each node u in $\mathcal{L}$ is a four-element tuple, $u = \ < lch, rch, PL, vec >$, where lch and rch store u's left and right child pointers, respectively, vec stores a keyword vector, and PL stores location vectors. The setting of u depends on its position in $\mathcal{L}$:

1) If u is a leaf node, then u corresponds to a pair $(V_{w_i}, PL_{i,j}) \in KL$, where $u.lch = u.rch = \varnothing$, $u.vec = V_{w_i}$ and $u.PL = PL_{i,j}$.
2) If u is an internal node, then u is constructed by its child nodes, where $u.lch$ and $u.rch$ store u 's left and right child pointers respectively, $u.vec = u.lch.vec \vee u.rch.vec$ and $u.PL = \varnothing$.

Algorithm 4: *GenBKFtree*(KL)

```
Input: the EPKI-index $KL$
Output: the BKFtree-index $\mathcal{L}$
1  Initialize two lists $L_1 = \varnothing$ and $L_2 = \varnothing$;
2  for each $(V_{w_i}, PL_{i,j}) \in KL$ do
3      Create a leaf node $u$;
4      $u.lch = u.rch = \varnothing$, $u.vec = V_{w_i}$, $u.PL = PL_{i,j}$;
5      Add $u$ into $L_1$;
6  while $|L_1| > 1$ do
7      while $|L_1| > 0$ do
8          if $|L_1| > 1$ then
9              Fetch the first two nodes $\{u_1, u_2\}$ from $L_1$;
10             Create a new parent node $u$ for $\{u_1, u_2\}$;
11             $u.lch = u_1$, $u.rch = u_2$, $u.PL = \varnothing$;
12             $u.vec = u_1.vec \vee u_2.vec$;
13             Add $u$ into $L_2$;
14         else if $|L_1| = 1$ then
15             Move the only one node left in $L_1$ to $L_2$;
16         else if $|L_1| = 0$ then
17             Move all the nodes of $L_2$ to $L_1$;
18     The BKFtree-index $\mathcal{L}$ is generated, the root of
        which is the only one node left in $L_1$;
19 return $\mathcal{L}$;
```

The construction of BKFtree-index is shown in Algorithm 4. The leaf nodes are generated from $(V_{w_i}, PL_{i,j}) \in KL$, then the index is constructed in a bottom-up way until there is one root node generated. Fig. 5 shows an example of BKFtree-index which is based on the EPKI-index in Fig. 4.

B. Design of EPSRQ+

We adopt the binary search tree to propose an optimized version of EPSRQ which is named EPSRQ+. Specifically, we need to improve the algorithms *GenIndex* and *Query* while other algorithms are kept the same. The improved algorithms are presented as follows:

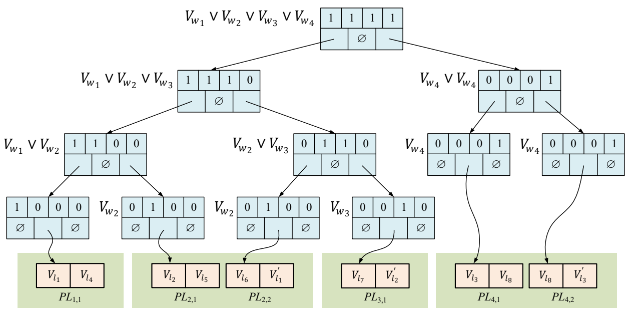


Fig. 5: An example of BKFtree-index

$\{\widetilde{\mathcal{L}}, \widetilde{D}\} \leftarrow$ ***GenIndex***$(\boldsymbol{W}, \boldsymbol{D}, \boldsymbol{sk}, \boldsymbol{\gamma})$. DO uses the algorithm *GenIndex* to generate the encrypted BKFtree-index by inputting the keyword dictionary W, the spatial-keyword dataset D, the partition threshold γ and the secret key set sk. DO first runs the algorithm *GenBKFtree* (Algorithm 4) to construct the BKFtree-index $\mathcal{L}$. Then, DO encrypts $\mathcal{L}$ and D which is the same as the original *GenIndex*.

After the BKFtree-index and spatial-keyword dataset are encrypted, DU outsources the encrypted data to CS.

Algorithm 5: *Query*$(\widetilde{D}, \widetilde{\mathcal{L}}, TD)$

```
Input: the encrypted spatial-keyword dataset $\widetilde{D}$, the
       encrypted BKFtree-index $\widetilde{\mathcal{L}}$, and the trapdoor
       $TD = \{\widetilde{V}_{W_Q}, \widetilde{V}_{\mathcal{R}}\}$
Output: the query result $R$
1  Initialize a set variable $S = \varnothing$;
2  SearchTree$(\widetilde{D}, \widetilde{\mathcal{L}}.root, TD, S)$;
3  $R = \bigcap_{CR_i \in S} CR_i$;
4  return $R$;
5  Function SearchTree$(\widetilde{D}, u, TD, S)$:
6      if $u \neq \varnothing$ then
7          if $u.\widetilde{vec} \cdot \widetilde{V}_{W_Q} \neq 0$ then
8              if $u$ is a leaf node then
9                  Initialize a candidate result set $CR_i$;
10                 for each $\widetilde{V}_{l_t} \in u.PL$ do
11                     if $\widetilde{V}_{l_t} \cdot \widetilde{V}_{\mathcal{R}} = 0$ then
12                         Add the encrypted item $(\widetilde{l}_t, \widetilde{W}_t)$
                            corresponding to $\widetilde{V}_{l_t}$ into $CR_i$;
13                 Add $CR_i$ into $S$;
14             else
15                 SearchTree$(\widetilde{D}, u.lch, TD, S)$;
16                 SearchTree$(\widetilde{D}, u.rch, TD, S)$;
```

$\boldsymbol{R} \leftarrow$ ***Query***$(\widetilde{\boldsymbol{D}}, \widetilde{\boldsymbol{\mathcal{L}}}, \boldsymbol{TD})$. Once CS receives a query trapdoor $TD = \{\widetilde{V}_{W_Q}, \widetilde{V}_{\mathcal{R}}\}$, CS runs Algorithm 5 to perform the spatial-keyword range query processing. CS first prunes the irrelevant subtree of the BKFtree-index to query the qualified tuples $(V_{w_i}, PL_{i,j}) \in KL$, then CS finds the locations in the query range in $PL_{i,j}$ to obtain the candidate query result set. The final query result is the intersection of the candidate result set. The query processing is described in Algorithm 5.

By using the BKFtree-index in the improved query algorithm, the subtrees having non-candidate keywords are pruned. The determination of candidate keywords is logarithmic to the number of posting lists. Assuming that the number of $PL_{i,j}$ is α and the length of each $PL_{i,j}$ is γ, the time complexity of the baseline query algorithm is $O(\alpha\cdot\gamma)$ while the optimized query algorithm is $O(log\alpha\cdot\gamma)$.

C. Dynamic Update

Similar to the dynamic update method in EPSRQ, EPSRQ+ also supports updates such as insert and delete.

Insert. We first construct a new EPKI-index KL^* for the items to be inserted. Then we construct a new BKFtree $\mathcal{L}^*$ for KL^* and merge two BKFtree trees. Finally we complete the insert operation.

Delete. When performing a delete operation, for each item to be deleted, similar to the query operation, we first locate it in $\mathcal{L}$ according to Algorithm 5 and then replace it with a phantom location vector.

VII. Security Analysis

In this section, we specifically give the security analysis of the proposed schemes. According to Definition 3, we will analyze the security of $Leak_{SP}$. Since $\mathcal{F}^{GK}$ is secure against λ, we mainly discuss $\mathcal{F}^{GI}$ and $\mathcal{F}^{GT}$.

Theorem 1. EPSRQ and EPSRQ+ can achieve the unlinkability, and the schemes are perfect $\mathcal{F}$-adaptively secure with the leakage functions $\mathcal{F}^{GI}$ and $\mathcal{F}^{GT}$.

Proof. We first prove the theorem with $\mathcal{F}^{GT}$. $\mathcal{A}$ has adaptive access to the encryption oracle and decryption oracle. A simulator $\mathcal{B}$ is constructed to interact with $\mathcal{A}$. The interactions between them are designed according to the indistinguishable experiment in Definition 1, which are shown as follows:

Init. $\mathcal{B}$ randomly generates the secret key $k=\{s,M_1,M_2\}$, where s is a m-dimentional vector and M_1 and M_2 are $m\times m$-dimensional invertible matrices.

Phase 1.

- **Encryption query.** $\mathcal{A}$ sends a query vector v_i to $\mathcal{B}$. $\mathcal{B}$ encrypts v_i as $\widetilde{v}_i$ and sends it to $\mathcal{A}$.
- **Decryption query.** $\mathcal{A}$ sends a query ciphertext $\widetilde{v}_i$ to $\mathcal{B}$. $\mathcal{B}$ decrypts $\widetilde{v}_i$ as v_i and sends it to $\mathcal{A}$.

Challenge. $\mathcal{A}$ first sends two vectors v_{c_0} and v_{c_1} to $\mathcal{B}$. $\mathcal{B}$ randomly chooses $b\in\{0,1\}$ and encrypts v_{c_b} as $\widetilde{v}_{c_b}$. Then $\mathcal{B}$ sends $\widetilde{v}_{c_b}$ to $\mathcal{A}$.

Phase 2.

- **Encryption query.** $\mathcal{A}$ sends a query vector v_i $(v_i\notin\{v_{c_0},v_{c_1}\})$ to $\mathcal{B}$. $\mathcal{B}$ encrypts v_i as $\widetilde{v}_i$ and sends it to $\mathcal{A}$.
- **Decryption query.** $\mathcal{A}$ sends an encrypted query vector $\widetilde{v}_i$ $(\widetilde{v}_i\neq\widetilde{v}_{c_b})$ to $\mathcal{B}$. $\mathcal{B}$ decrypts $\widetilde{v}_i$ as v_i and sends it to $\mathcal{A}$.

Guess. $\mathcal{A}$ outputs a guess b' of b. $\mathcal{B}$ outputs true if $b'=b$, otherwise, it outputs false.

In the above interactions, we assume that the number of encryption and decryption queries in **Phase 1** are q_{e_1} and q_{d_1}, respectively; and the number of encryption and decryption queries in **Phase 2** are q_{e_2} and q_{d_2}, respectively. The total number of encryption and decryption queries in **Phase 1** and **Phase 2** is denoted as q, $q=q_{e_1}+q_{d_1}+q_{e_2}+q_{d_2}$, and we have $q<m-2$ according to Definition 1.

Afterwards, we prove that $Adv_{\mathcal{A}}(k)=|Pr[b'=b]-\frac{1}{2}|=0$ holds. Suppose that the query vector set $V=\{v_1,v_2,\cdots,v_q\}$ and the secret key $\{s,M_1,M_2\}$, each query vector $v_i\in V$ is a m-dimentional vector, s is a m-dimentional vector and M_1 and M_2 are $m\times m$-dimensional invertible matrices. The encrypted vector of v_i is $\widetilde{v}_i=\{M_1^{-1}v_i', M_2^{-1}v_i''\}$. Thus, we have that for any m-dimensional vector $v_j\in\mathbb{Z}_p^m$, there exists a secret key $k^*=\{s^*,M_1^*,M_2^*\}$ and $k^*\neq k$, such that

$$\forall v_i\in V, \widetilde{v}_i=Enc_{k^*}(v_i) \tag{25}$$

and for v_{c_b},

$$\widetilde{v}_{c_b}=Enc_{k^*}(v_j). \tag{26}$$

The proof of the existence of k^* is given as follows. If k^* exists, it should satisfy the following $2qm+2m$ equations:

$$\begin{cases}\cdots\\ M_1^{-1}v_i'=(M_1^*)^{-1}v_i'\\ M_2^{-1}v_i''=(M_2^*)^{-1}v_i''\\ \cdots\\ M_1^{-1}v_{c_b}'=(M_1^*)^{-1}v_j'\\ M_2^{-1}v_{c_b}''=(M_2^*)^{-1}v_j''\end{cases}. \tag{27}$$

Since the vectors $\{v_i{}',v_i{}''\}$, $\{v_j{}',v_j{}''\}$, $\{v_{c_b}{}',v_{c_b}{}''\}$ and the matrices $\{M_1,M_2\}$ are known, $\{M_1^*,M_2^*\}$ is unknown and contains $2m^2$ unknown elements, and $2m^2>2qm+2m$, we can deduce that there exists infinite $\{M_1^*,M_2^*\}$ satisfying Eq. 27, and thus k^* exists. Therefore, the probability of decrypting the encrypted vector $\widetilde{v}_{c_b}$ as any vector is $\frac{1}{q}$, i.e.,

$$Pr[Dec(\widetilde{v}_{c_b})=v_{c_b}]=Pr[Dec(\widetilde{v}_{c_b})=v_j]=\frac{1}{q}, \tag{28}$$

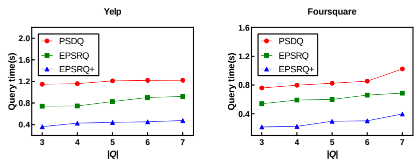
and thus, $\mathcal{A}$ can't distinguish the decryption results of $\widetilde{v}_{c_b}$, which means that $Adv_{\mathcal{A}}(k)=|Pr[b'=b]-\frac{1}{2}|=0$ holds. According to Definition 1 and 2, the trapdoor generation algorithm can achieve perfect indistinguishability, and the unlinkability is proved.

The encryption method used for the generated index is similar to a trapdoor. Therefore, the proof mentioned above also holds for $\mathcal{F}^{GI}$.

In conclusion, according to the above proofs, we have that the schemes Π can achieve unlinkability, and Π is perfect $\mathcal{F}$-adaptively secure. ■

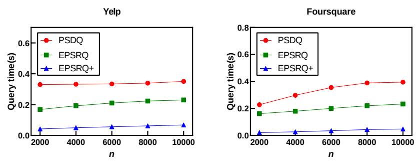
VIII. Performance Evaluation

In this section, we first introduce the dataset and parameter settings. Then we perform a comprehensive evaluation for EPSRQ, EPSRQ+ and PSDQ [19]. All the above evaluations are realized in Python. To evaluate the performance of our schemes, we utilize the real datasets, Yelp academic dataset business [27] and the Foursquare dataset of NYC [28].



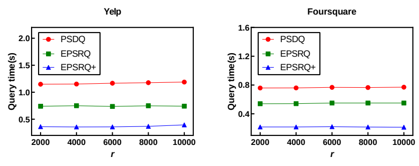
(a) (b)

Fig. 6: Query time versus $|Q|$



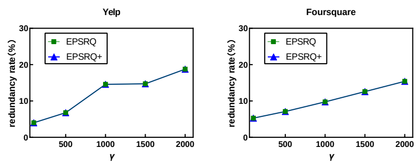
(a) (b)

Fig. 7: Query time versus n



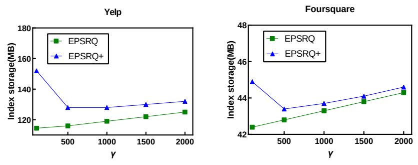
(a) (b)

Fig. 8: Query time versus r



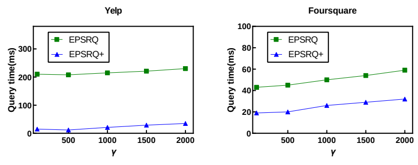
(a) (b)

Fig. 9: Redundancy rate versus γ



(a) (b)

Fig. 10: Index storage versus γ



(a) (b)

Fig. 11: Query time versus γ

The default parameters are shown in Table I, where n is the number of spatial-keyword items in the dataset, m is the size of the keyword dictionary, γ is the partition threshold, r is the query range, $|Q|$ is the number of query keywords.

TABLE I: Default parameter settings

Dataset	n	m	γ	r	$\|Q\|$
Yelp	150346	1311	500	200	3
Foursquare	37994	250	500	200	3

A. *Query Time Evalutation*

We evaluate the query time under different parameters, and the results are shown in Fig. 8-12.

Query time cost versus $|Q|$. Fig. 6 shows that as the number of query keywords $|Q|$ increases, the query time of PSDQ, EPSRQ and EPSRQ+ increases. The reason for this is that as $|Q|$ increases, more tree nodes will be accessed, and the more vectors that are performed in the scalar products, thus, the time costs of the three schemes increase as $|Q|$ grows. The experimental results show that EPSRQ+ performs best.

Query time cost versus n. Fig. 7 shows that as the number of spatial-keyword items increases, the query time of the three schemes all increases simultaneously. The reason for this is that with the growth of n, the number of queried locations increases as n increases in the three schemes, which will allow more time for performing scalar product calculations. Additionally, in three schemes, the growth rate of time cost in EPSRQ+ is obviously smaller than that in the other schemes.

Query time cost versus r. Fig. 8 shows that as the query range r increases, the query time of PSDQ, EPSRQ and EPSRQ+ tends to be stable. This is because r only affects the vector in the trapdoor TD, which has no effect on the computation of the scalar product during the query processing. Therefore, with different parameters r, the query time fluctuates little and the overall trend is smooth.

B. *Partition Threshold Setting Evaluation*

The partition threshold γ is an important parameter that directly affects the index storage, the redundancy rate and the query time cost. We conduct experiments on the partition threshold setting in this subsection.

Redundancy rate versus γ. Figure 9 shows that as γ increases, the redundancy rate of the index in EPSRQ and EPSRQ+ all increase simultaneously, where $Redundancy\_rate = \frac{\sum |PL_{i,j}| - n}{n} \times 100\%$, $PL_{i,j}$ contains the location vectors corresponding to w_i. The reason is that as γ increases, the number of phantom location vectors in the index also increases.

Index storage versus γ. Fig. 10 shows that as γ increases, the index storage of EPSRQ grows linearly and the index storage of EPSRQ+ initially decreases and then increases. The reason is that in EPSRQ, the index size increases as γ increases due to the growth in the number of locations. However, in EPSRQ+, despite the small number of locations, the smaller γ is, the more the number of posting lists is, and therefore the number of leaf nodes of the BKFtree becomes larger.

Query time cost versus γ. Fig. 11 shows that as γ increases, the overall trend of the query time for EPSRQ and EPSRQ+ grows linearly. The reason is that as γ increases, the number of locations in PL also increases, which will allow more time for performing scalar product calculations.

IX. Conclusion

In this paper, we propose the efficient privacy-preserving spatial-keyword range query processing in the cloud. Keywords, locations and ranges in the spatial keyword data are vectorized to hide the spatial information. An equal partition-based keyword-location inverted index is constructed on the vectors of spatial-keyword data, and a baseline privacy-preserving spatial-keyword range query scheme is proposed. To improve query efficiency, a novel binary keyword filtering tree index is designed. By using the index, an efficient spatial-keyword range query scheme is proposed. The proposed schemes can protect the privacy of the spatial-keyword data from the curious cloud service provider and preserve the trapdoor unlinkability. Experiments conducted on real-world datasets show that the proposed schemes have better performance in query efficiency.

References

[1] Q. Tong, Y. Miao, H. Li, X. Liu, and R. Deng, “Privacy-preserving ranked spatial keyword query in mobile cloud-assisted fog computing,” *IEEE Transactions on Mobile Computing*, 2021.

[2] C. Zhang, L. Zhu, C. Xu, J. Ni, C. Huang, and X. Shen, “Location privacy-preserving task recommendation with geometric range query in mobile crowdsensing,” *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4410–4425, 2021.

[3] F. Song, Z. Qin, D. Liu, J. Zhang, X. Lin, and X. Shen, “Privacy-preserving task matching with threshold similarity search via vehicular crowdsourcing,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 7, pp. 7161–7175, 2021.

[4] V. K. Yadav, S. Verma, and S. Venkatesan, “Linkable privacy-preserving scheme for location-based services,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7998–8012, 2021.

[5] Y. Ren, X. Li, Y. Miao, R. Deng, J. Weng, S. Ma, and J. Ma, “Distpreserv: maintaining user distribution for privacy-preserving location-based services,” *IEEE Transactions on Mobile Computing*, 2022.

[6] Z. Chen, J. Nie, Z. Li, W. Susilo, and C. Ge, “Geometric searchable encryption for privacy-preserving location-based services,” *IEEE Transactions on Services Computing*, 2023.

[7] J. Liang, Z. Qin, L. Xue, X. Lin, and X. Shen, “Efficient and privacy-preserving decision tree classification for health monitoring systems,” *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 528–12 539, 2021.

[8] H. Zhu, R. Lu, C. Huang, L. Chen, and H. Li, “An efficient privacy-preserving location-based services query scheme in outsourced cloud,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7729–7739, 2015.

[9] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, “Enabling efficient and geometric range query with access control over encrypted spatial data,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 870–885, 2018.

[10] X. Lei, A. X. Liu, R. Li, and G.-H. Tu, “Seceqp: A secure and efficient scheme for sknn query problem over encrypted geodata on cloud,” in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 662–673.

[11] R. Guo, B. Qin, Y. Wu, R. Liu, H. Chen, and C. Li, “Luxgeo: efficient and secure enhanced geometric range queries,” *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[12] F. Wang, H. Zhu, G. He, R. Lu, Y. Zheng, and H. Li, “Efficient and privacy-preserving arbitrary polygon range query scheme over dynamic and time-series location data,” *IEEE Transactions on Information Forensics and Security*, 2023.

[13] N. Cui, J. Li, X. Yang, B. Wang, M. Reynolds, and Y. Xiang, “When geo-text meets security: privacy-preserving boolean spatial keyword queries,” in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1046–1057.

[14] X. Wang, J. Ma, X. Liu, R. H. Deng, Y. Miao, D. Zhu, and Z. Ma, “Search me in the dark: Privacy-preserving boolean range query over encrypted spatial data,” in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2253–2262.

[15] N. Cui, X. Yang, Y. Chen, J. Li, B. Wang, and G. Min, “Secure boolean spatial keyword query with lightweight access control in cloud environments,” *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9503–9514, 2021.

[16] F. Song, Z. Qin, L. Xue, J. Zhang, X. Lin, and X. Shen, “Privacy-preserving keyword similarity search over encrypted spatial data in cloud computing,” *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 6184–6198, 2021.

[17] Y. Yang, Y. Miao, Z. Ying, J. Ning, X. Meng, and K.-K. R. Choo, “Privacy-preserving threshold spatial keyword search in cloud-assisted iiot,” *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16 990–17 001, 2021.

[18] Y. Yang, Y. Miao, K.-K. R. Choo, and R. H. Deng, “Lightweight privacy-preserving spatial keyword query over encrypted cloud data,” in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 392–402.

[19] Y. Miao, Y. Yang, X. Li, L. Wei, Z. Liu, and R. H. Deng, “Efficient privacy-preserving spatial data query in cloud computing,” *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[20] Q. Meng, J. Weng, Y. Miao, K. Chen, Z. Shen, F. Wang, and Z. Li, “Verifiable spatial range query over encrypted cloud data in vanet,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 12 342–12 357, 2021.

[21] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, “Privacy-preserving multi-keyword ranked search over encrypted cloud data,” *IEEE Transactions on parallel and distributed systems*, vol. 25, no. 1, pp. 222–233, 2013.

[22] D. Das, R. Amin, and S. Kalra, “Algorithm for multi keyword search over encrypted data in cloud environment,” in *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2020, pp. 733–739.

[23] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, “Achieving efficient and privacy-preserving set containment search over encrypted data,” *IEEE Transactions on Services Computing*, vol. 15, no. 5, pp. 2604–2618, 2021.

[24] Y. Miao, W. Zheng, X. Jia, X. Liu, K.-K. R. Choo, and R. H. Deng, “Ranked keyword search over encrypted cloud data through machine learning method,” *IEEE Transactions on Services Computing*, vol. 16, no. 1, pp. 525–536, 2022.

[25] Q. Zhou, H. Dai, Z. Hu, Y. Liu, and G. Yang, “Accuracy-first and efficiency-first privacy-preserving semantic-aware ranked searches in the cloud,” *International Journal of Intelligent Systems*, vol. 37, no. 11, pp. 9213–9244, 2022.

[26] S. K. Kermanshahi, S.-F. Sun, J. K. Liu, R. Steinfeld, S. Nepal, W. F. Lau, and M. H. A. Au, “Geometric range search on encrypted data with forward/backward security,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 698–716, 2020.

[27] “Yelp,” https://www.yelp.com/dataset, 2014.

[28] “Foursquare,” https://sites.google.com/site/yangdingqi/home/foursquare-dataset, 2013.