Privacy Protection Space Keyword Range Query Technology

Li Zhou
1023041116
Nanjing University of Posts and Telecommunications
Jiangsu, Nanjing China

***Abstract*—With the rapid development of location-based services in mobile internet, more and more data owners choose to outsource their spatial text data to cloud service providers for the flexibility and cost savings that come with querying. However, directly outsourcing such data services to untrusted cloud service providers may pose serious privacy issues. Therefore, an effective cloud-based encrypted search scheme is needed to address this problem. The main work of this paper includes:**

A privacy-preserving spatial keyword range query method based on KLIAT-index is proposed. By introducing an inverted index structure and AVL tree, a keyword-location inverted AVL tree (KLIAT-index) mechanism is designed, and an index construction algorithm is proposed. The index data is encrypted using symmetric encryption, 0-1 encoding, ASPE, HMAC, and other methods to improve security. Based on the encrypted index, a privacy-preserving spatial keyword range query method is proposed. Experiments show that this query method is more efficient than existing similar works.

I. INTRODUCTION

In recent years, with the continuous development of big data technology, more and more people have begun to pay attention to cloud computing, which means storing big data through cloud servers, achieving data storage management and corresponding computing. Through this approach, businesses or individuals can save storage space and computational costs. In location information queries, storing spatial text data of a location on a cloud server is beneficial for efficient user queries. However, even with various security measures in place on current cloud servers, storing data in the cloud still faces privacy breaches. How to conduct secure queries while ensuring data privacy is currently a concern for people. This article mainly studies how to achieve spatial keyword range queries for privacy protection.

This article proposes a new privacy preserving spatial keyword query scheme. Firstly, use an inverted index to map each keyword to the corresponding position point list, and then construct a tree index structure for the longitude of the keywords and position points separately. The storage of location points utilizes AVL trees to improve retrieval efficiency by constructing a tree index based on longitude, and performs 0-1 encoding and HMAC on the data to achieve privacy preserving spatial keyword queries. Our main contributions to work are as follows:

(1) By constructing an inverted index and using keywords to filter location points before querying the range.

(2) Using AVL trees to construct indexes for longitude coordinates to improve retrieval efficiency, and then performing latitude traversal queries on the filtered results to obtain the final result.

(3) Using 0-1 encoding and HMAC to store and encrypt longitude and latitude coordinates, data queries can be performed under the premise of privacy protection.

(4) We provide security analysis and experiments to demonstrate the security and feasibility of our solution.

II. PRELIMINARY KNOWLEDGE

A. invert index

Inverted index is a way to retrieve documents by utilizing their attribute values. Inverted indexes are commonly used in information retrieval systems to quickly locate documents containing specific attribute values. In the inverted index, each attribute value has a corresponding document list, which contains the identifiers of all documents containing that attribute value. By using inverted indexes, a list of documents containing query attribute values can be quickly found. For example, there are 8 location points, namely $\{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8\}$, each corresponding to a keyword. Extract keywords and map each keyword to its corresponding set of positional points to generate an inverted index. The specific construction process is shown in Figure 1:

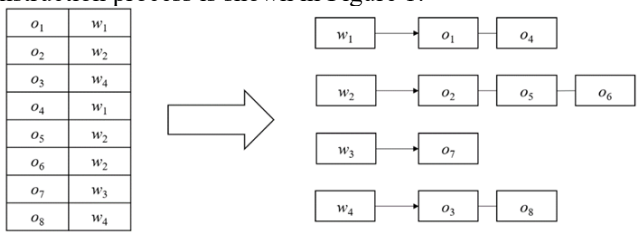

Fig. 1. invert index.

In this article, the inverted index method is used to map all keywords to the corresponding set of positional points, and the

first round of filtering is performed on positional points using keywords to improve retrieval efficiency.

B. 0-1 encoding

0-1 encoding is an encoding method that compares the size of data after binary encoding, and has privacy protection capabilities. The specific encoding method is as follows:

Let x be an integer containing w binary bits, where $x = b_1 b_2 \dots b_{w-1} b_w (b_i \in \{0,1\}, 1 \le i \le w)$. The 0 encoding set $ZC(x)$ and the 1 encoding set $OC(x)$ obtained by 0-1 encoding x are:

$$ZC(x) = \{b_1 b_2 \cdots b_{i-1} 1 | b_i = 0, 1 \le i \le w\}$$
$$OC(x) = \{b_1 b_2 \cdots b_i | b_i = 1, 1 \le i \le w\}$$

Among them, $b_1 b_2 \dots b_{w-1} b_w$ is the encoding after converting x to binary, where b_1 is the highest bit and b_w is the lowest bit.

For two integers x and y containing w binary bits, 0 encoding and 1 encoding are performed respectively. If and only if there is no intersection between $ZC(x)$ and $OC(y)$, $x \geqslant y$ holds; When there is a non empty intersection between ZC (x) and OC (y), x<y holds, which satisfies formulas:

$$x \ge y \leftrightarrow ZC(x) \cap OC(y) = \emptyset$$
$$x < y \leftrightarrow ZC(x) \cap OC(y) \neq \emptyset$$

The above process provides a method for determining the size relationship between two numerical values x and y, that is, by determining whether their 0 encoding set and 1 encoding set intersect. For example, encoding 4 and 8 with 8-bit 0 and 1 respectively yields $ZC(4)$={1, 01, 001, 0001, 00001, 0000011, 00000101},$OC(8)$={00001}.Since $ZC(4) \cap OC(8) = \{00001\} \neq \emptyset$, according to formula (3.2), 4<8 can be obtained. In this article, the latitude and longitude information of the location is stored in a 1 encoding. After converting the query data to a 0 encoding, it is compared with the stored 1 encoding to determine whether the location point is within the query range.

C. Asymmetric scalar product preserving encryption

Asymmetric Scalar Product Preserving Encryption (ASPE) is an asymmetric encryption scheme that supports preserving scalar product operations. In this encryption scheme, the scalar product of two vectors in the ciphertext can be calculated without exposing the plaintext, which can play a very important role in vector operations. Therefore, ASPE is suitable for secure computing in cloud computing and big data scenarios. In these scenarios, data is usually stored in the cloud, while computing tasks are initiated by the client and performed in the cloud. The cloud can return the calculation results to the user in a ciphertext state.

The specific implementation process of ASPE is as follows:

$\boldsymbol{ASPE.KeyGen(1^{\lambda}) \rightarrow sk}$: Given a security parameter λ, Generate key sk={s, M_1, M_2} through algorithm, where s is a random *d-dimensional* bit vector, and M_1 and M_2 are two random $d \times d$ dimensional invertible matrix.

$\boldsymbol{ASPE.Enc(p, sk) \rightarrow C}$: Given the key sk and vector p, encrypt p into C=$(M_1^T p', M_2^T p'')$ according to the algorithm, where p is divided into p' and p'' using the bit vector s in the key using formula.

$$\begin{cases} p'[k] = p''[k] = p[k], & if\ s[k] = 0; \\ p'[k] + p''[k] = p[k], & if\ s[k] = 1. \end{cases}$$

$\boldsymbol{ASPE.TrapGen(q, sk) \rightarrow T_Q}$: Given the key sk and vector q, encrypt q into $T_Q = (M_1^{-1} q', M_2^{-1} q'')$ according to the algorithm, where q is divided into q' and q'' using the bit vector s in the key using formula.

$$\begin{cases} q'[k] + q''[k] = q[k], & if\ s[k] = 0; \\ q'[k] = q''[k] = q[k], & if\ s[k] = 1. \end{cases}$$

$\boldsymbol{ASPE.Query(C, T_Q) \rightarrow p^T \cdot q}$**:** The result obtained by calculating C and T_Q using formula after encryption is still $p^T \cdot q$

$$C^T \cdot T_Q = ((p')^T M_1) \cdot (M_1^{-1} q') + ((p'')^T M_2) \cdot (M_2^{-1} q'')$$
$$= p^T \cdot q$$

III. Model Definition and Problem Description

A. System Model

This article proposes a scheme for keyword position inverted AVL tree indexing mechanism, involving an honest and curious security model. The entire retrieval system is divided into three parts: data owner DO (Date Owner), cloud server CS (Cloud Server), and data user DU (Date User). The relationship between the three is shown in Figure 2:

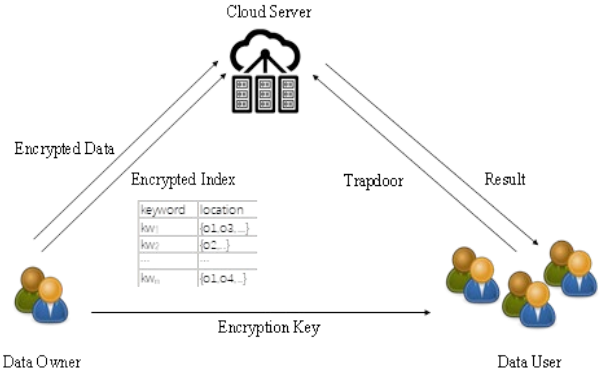

Fig. 2. System Model.

The role of each entity is as follows:

(1) Data Owner: The data owner first generates inverted indexes for keywords and location points in the dataset, then constructs indexes and encrypts spatial data objects, and outsources them to cloud servers.

(2) Data user: The user encrypts the query content into trapdoors and then submits the trapdoors to the cloud server.

(3) Cloud servers: Cloud servers have unlimited storage and computing capabilities. It can store encrypted objects and indexes outsourced by data owners and provide query services to users.

The main process of the system is as follows: after the data owner generates an inverted index, they construct a tree index for the keywords and location point sets, and then encrypt the content of the document and outsource it to the cloud server along with the generated inverted index. The user is authorized by the owner to obtain the encryption key. When a user requests a query, encrypt the query content as a trapdoor and submit the trapdoor to the cloud server. After receiving the trapdoor, the cloud server finds a result that meets the user's query

requirements and returns the result to the data user.

B. Threat Model

In cloud environments, servers are usually considered "honest and curious". In the "honest and curious" model, cloud servers strictly adhere to requirements to implement algorithms and provide services to users, but also curiously capture user stored data or query information. We assume that the user is trustworthy and will not communicate with other users or servers, and assume that the transmission channel is secure. The focus of the model design in this article is to achieve privacy protection for spatial keyword queries in a cloud environment based on the "honest and curious" threat model. In this model, we should achieve data privacy protection, including the location and keyword content in the dataset, index, and query. Cloud servers cannot obtain real plaintext information from encrypted databases, trapdoors, or indexing processes.

C. Problem description

Definition 3.1 Location Document: Let the spatial keyword location document be $D=\{O_1, O_2, \ldots, O_n\}$, where each O_i in D has a corresponding tuple (id_i, x_i, y_i, W_i), where id_i represents the position point, x_i, y_i represents the latitude and longitude coordinates of the position; $W_i=\{w_1, w_2, \ldots, w_t\}$ represents the keyword set at that location.

Our work mainly focuses on unfolding a two-dimensional geographic space composed of longitude and dimensions. Due to the fact that latitude and longitude coordinates represent positions on the Earth's sphere, they are not equidistant on the Earth's surface. In latitude and longitude coordinates, the length of a latitude degree is longer at the equator than at the poles, and the length of a longitude degree depends on latitude. This non-linear relationship makes it difficult to calculate the distance between two points using a simple distance formula. Therefore, when performing range queries, it is not possible to directly perform range queries on longitude and latitude coordinates. We use the geodetic coordinate system to convert longitude and latitude coordinates into a plane coordinate system, which can accurately describe the position of the Earth's surface and all are expressed in integer form, that is, (xi, yi) are integers and can be directly used for range queries.

Definition 3.2 Search Request: The search request is $Q=\{L_Q, W_Q, R\}$, where L_Q represents the current coordinates, including (x_Q, y_Q), W_Q represents the keyword set for the query, and R represents the query range.

Definition 3.3 Search Trapdoor: Before the data owner outsources dataset D to a cloud server, it is necessary to first establish an index $\mathcal{L}$ on the dataset and encrypt it, represented as $\mathcal{L}^*$, and encrypt dataset D, represented as $D^*=\{O_1^*, O_2^*, \cdots, O_n^*\}$. At the same time, the data owner needs to encrypt the user's query request Q to generate a trapdoor, represented as $TD=\{R_Q^*, W_Q^*\}$, where R_Q^* is the range of the query matrix obtained based on the current coordinates and query range.

Definition 3.4 Privacy preserving Spatial Keyword Query: Given an encrypted dataset D^*, an encrypted index $\mathcal{L}^*$, and a trapdoor TD, the cloud server can use TD to retrieve $\mathcal{L}^*$ from D^* based on encrypted data, and retrieve results from Γ^*。

The privacy protection space keyword query includes the following four algorithms:

(1) $\boldsymbol{Setup}(\mathbf{1}^{\boldsymbol{\lambda}}) \rightarrow \boldsymbol{sk}$: Given a security parameter λ, This algorithm generates keys $sk=\{s, M_1, M_2, k\}$, where s is a random *d-dimensional* bit vector and M_1, M_2 are two randomly generated $d \times d$-*dimensional* reversible matrix, k is used to generate HMAC for the range of queries.

(2) $\boldsymbol{IndexBuild}(\boldsymbol{sk}, \boldsymbol{D}) \rightarrow \boldsymbol{\mathcal{L}^*}$: The data owner uses the received *sk* to construct an index on dataset D and encrypt it to achieve privacy protected range queries. Afterwards, upload $\mathcal{L}^*$ to the cloud.

(3) $\boldsymbol{TrapdoorGen}(\boldsymbol{sk}, \boldsymbol{Q}) \rightarrow \boldsymbol{TD}$: Generate a query trapdoor *TD* using the user's query request and key *sk*.

(4) $\boldsymbol{Search}(\boldsymbol{\mathcal{L}^*}, \boldsymbol{TD}, \boldsymbol{D^*}) \rightarrow \boldsymbol{\Gamma^*}$: It is a deterministic algorithm implemented on the cloud server side. It takes the security index $\mathcal{L}^*$, valve *TD*, and encrypted dataset D^* as inputs and outputs the query results Γ^*。

IV. Privacy preserving spatial keyword range query based on KLIAT index

A. KLIAT-index indexing mechanism

Definition 4.1 Keyword Set: The keyword set $W=\{w_1, w_2, \ldots, w_n\}$ contains all the keywords in the positional document, that is, all the O_i contained in the positional document D in Definition 3.1 W_i union composition.

Definition 4.2 Keyword Location Inverted AVL Tree Index: Construct a mapping relationship between keywords and location points based on the coordinates of the location points in the keyword set W and the location document D, and store it in the inverted index. The position point set corresponding to each keyword $w_i \in W$ is used to construct an AVL tree based on longitude, forming a binary (w_i, N_i), where Ni represents the root node of the AVL tree. The construction results are shown in Figure 3:

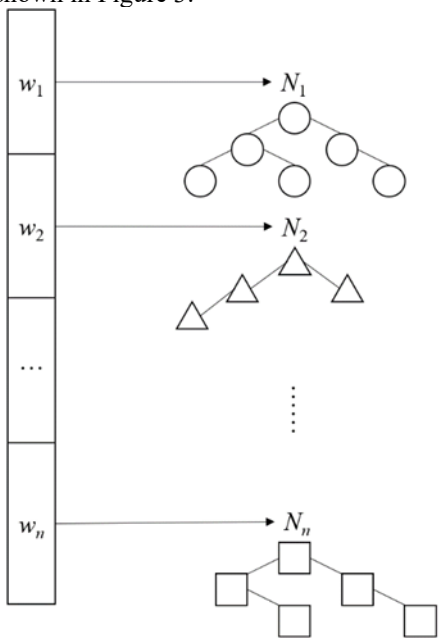

Fig. 3. KLIAT-index

Definition 4.3 Keyword Search Tree: In order to facilitate the retrieval of keywords, we have established a keyword search tree for the keywords. Generate an *n-dimensional* vector for each keyword w_i in the keyword set $W=\{w_1, w_2, \ldots, w_n\}$ (γ_1, γ_2, …, γ_n)Where n is the number of keywords, $i \in [1, n]$. The conditions that are met are shown in formula :

$$\begin{cases}\gamma_i = 0, w_i \notin W \\ \gamma_i = 1, w_i \in W\end{cases}$$

After generating a vector for each keyword, a keyword search tree is built from bottom to top using the vector as the leaf node, where the father node is the result of OR operation between two child nodes. For example, let $W=\{w_1, w_2, w_3, w_4\}$. Taking this as an example, the process of building a keyword tree is shown in Figure :

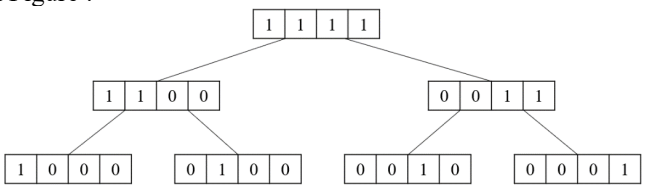

Fig. 4. Keyword search tree

B. Encrypted index generation

To ensure the security of the query process and achieve privacy protected scope queries, we need to encrypt the constructed index. The encryption process is divided into two parts, namely encryption of keywords and encryption of location points.

To ensure that the cloud server can perform queries in the keyword search tree without exposing plaintext information, the key $k=\{s, M_1, M_2\}$ is used to encrypt the nodes of each tree and the keyword W_Q in the trapdoor before uploading the index to the cloud server. This allows for the calculation of the inner product of two vectors without compromising privacy. The specific encryption process is as follows:

(1) Let the data stored in each node in the tree be a vector p. Encrypt p with keys $k=\{s, M_1, M_2\}$ to $C = (M_1^T p', M_2^T p'')$, where p is divided into $p'$and p'' using the bit vector s in the key using formula

$$\begin{cases}p'[k] = p''[k] = p[k], & if\ s[k] = 0; \\ p'[k] + p''[k] = p[k], & if\ s[k] = 1.\end{cases}$$

(2) Convert the keyword set W_Q in the query request into vector form and record it as q, and encrypt q to $W_Q^* = (M_1^{-1}q', M_2^{-1}q'')$ using the key $k=\{s, M_1, M_2\}$, where q is divided into $q'$and q'' using the bit vector s in the key using formula .

$$\begin{cases}q'[k] + q''[k] = q[k], & if\ s[k] = 0; \\ q'[k] = q''[k] = q[k], & if\ s[k] = 1.\end{cases}$$

(3) The result obtained by calculating C and W_Q^* through formula after encryption is still is still $p^T \cdot q$.

$$C^T \cdot W_Q^* = ((p')^T M_1) \cdot (M_1^{-1}q') + ((p'')^T M_2) \cdot (M_2^{-1}q'') = p^T \cdot q$$

After encrypting the data, the cloud server can calculate and query the results in a ciphertext state.

For the coordinates in the location point set, we use 0-1 encoding to encrypt them.

Due to the reversible nature of 0-1 encoding, HMAC needs to be used to eliminate this feature when storing 0-1 encoded content.

HMAC is a message authentication code (MAC) that requires the use of an encrypted hash function and a key to generate an authentication code. HMAC is commonly used to verify the authenticity and integrity of messages and data transmitted over the network or stored in databases.

Generating HMAC requires applying hash functions (such as SHA-256) to combine messages and keys. The result is a fixed size hash value that is unique for both the message and the key. This hash value is HMAC.

HMAC provides a method for verifying message integrity without storing the message itself. Just store HMAC. If the query user has the same key as the sender, the same hash function can be used to calculate HMAC and compare it with the stored HMAC. If two HMACs match, the message can be considered true and tampered with. If two HMACs do not match, the query request may have been modified during transmission, or the key may be incorrect. Therefore, after storing leaf nodes with HMAC, since the comparison method of 0-1 encoding is obtained by calculating the intersection, if the key is incorrect or the message is tampered with during transmission, there will be no intersection in the final query, and therefore no corresponding result will be returned.

Here, we encode the coordinates of the position point set to 1, which is used to intersect with the 0 encoding range in the subsequent query to obtain the query result.

C. Data preprocessing

(1) Based on the spatial dataset D={(W1, l1), (W2, l2) , · · · , (Wm, lm)} $D=\{(W_1, l_1), (W_2, l_2) , \cdots , (W_m, l_m)\}$, construct a keyword dictionary $W=\{w_1, w_2, \ldots, w_n\}$, where $l_i=(x_i, y_i)$ is a two-dimensional spatial location point, and W_i is a set of keywords that describe the location;

(2) Based on dictionary W and spatial dataset D, construct a keyword position set binary sequence $L=\{L_1, L_2, \ldots, L_n\}$, where $L_i=(w_i, Loc_i)$ is the keyword position set binary, and Loc_i represents the set of positions containing w_i in the position description keyword set, i.e. $Loc_i = \{l_j \mid (W_j, l_j) \in D \wedge w_i \in W_j\}$;

(3) For each keyword position set binary L_i in L, based on the partition parameters τ Split the position set Loc_i in L_i and perform isometric processing on the split result to generate $h = \left\lceil \frac{|Loc_i|}{\tau} \right\rceil$Key words - position set L_i'=$\{L_{i,1}, L_{i,2}, \ldots, L_{i,h}\}$, where $L_{i,j}=(w_i, Loc_{i,j})$, $Loc_{i,j}$, $Loc_{i,j}$ contains τ A collection of location points. After splitting, a keyword position set binary sequence $L' = \bigcup_{i=1}^{n} L_i'$ is formed with an equal number of position points;

(4) For each keyword position set $L_{i,j}=(w_i, Loc_{i,j})$ in the keyword position set binary L_i' with an equal number of position points, construct the corresponding AVL search tree $T_{i,j}$ based on the horizontal or vertical coordinates of each position point in $Loc_{i,j}$, and generate the keyword search tree binary (w_i, $T_{i,j}$). Finally, the keyword search tree binary

sequence $\mathcal{L} = \{(w_i, T_{i,j}) | w_i \in W \wedge L_{i,j} \in L'\}$ is generated, where $T_{i,j}$ are the AVL search trees corresponding to $L_{i,j}$;

(5) Using 0-1 encoding and HMAC, encrypt and sort each keyword search tree binary (w_i, $T_{i,j}$) in L to generate an encrypted keyword search tree binary sequence $\tilde{\mathcal{L}}$, This sequence is an encrypted index that supports efficient privacy preserving spatial keyword queries.

Furthermore, in step (3), based on the partition parameters τ The process of splitting L_i=(w_i, Loc_i) into equal lengths includes the following steps:

(3.1) If the keyword $w_i \in W$ The set of position points corresponding to i Loc_i contains a number of position points less than or equal to the partition parameters τ, Construct $\tau - |Loc_i|$ virtual locations to Loc_i to obtain a set of locations Loci, 1. At this point, the keyword position set formed by L_i after splitting is L_i'={$L_{i,1}$}, where $L_{i,1}$=(w_i, $Loc_{i,1}$);

(3.2) If the keyword $w_i \in W$ The set of position points corresponding to Loc_i contains more position points than the partition parameters τ, Split Loc_i into $h = \left\lceil \frac{|Loc_i|}{\tau} \right\rceil$ A subset of positions G={G_1, G_2, …, G_h}, and in order to ensure minimal similarity in filling virtual positions, the absolute value of the difference between the number of positions contained in any two subsets should not exceed 1. Construction $(h * \tau - |Loc_i|)$ virtual positions, for each subset G_j of G, if the number of positions contained in G_j is less than τ, Then the $\tau - |Loc_{i,j}|$ virtual locations added to G_j. Make each G_j contain τ Location points. At this point, the keyword position set formed by L_i after splitting is L_i'={ $L_{i,1}$, $L_{i,2}$, …, $L_{i,h}$ }, where any keyword position set $L_{i,j}$=(w_i, $Loc_{i,j}$), $Loc_{i,j}$=G_j.

Furthermore, the process of generating the keyword search tree binary sequence L in step (4) includes the following steps:

(4.1) For each keyword in L' position set $L_{i,j}$=(w_i, $Loc_{i,j}$), construct the corresponding AVL search tree $T_{i,j}$ based on the horizontal or vertical coordinates of each position point in $Loc_{i,j}$. Since the length of $Loc_{i,j}$ is τ, So the size and height of the constructed AVL search tree are the same;

(4.2) For each keyword w_i and its corresponding AVL search tree $T_{i,j}$, generate a keyword search tree binary (w_i, $T_{i,j}$), which means that each corresponding search tree { $T_{i,1}$, $T_{i,2}$, …, $T_{i,j}$ } can be obtained from the keyword wi. The final generated keyword search tree binary sequence $\mathcal{L} = \{(w_i, T_{i,j}) | w_i \in W \wedge L_{i,j} \in L'\}$, where $T_{i,j}$ are the AVL search trees corresponding to $L_{i,j}$.

Furthermore, in step (5), each keyword search tree binary (w_i, $T_{i,j}$) in $\mathcal{L}$ is encrypted and sorted to obtain the encrypted keyword search tree binary sequence $\tilde{\mathcal{L}}$, The specific process includes the following steps:

(5.1) For each keyword in $\mathcal{L}$ search tree binary (w_i, $T_{i,j}$), first encrypt the AVL search tree $T_{i,j}$ using 0-1 encoding and HMAC to generate an encrypted AVL search tree $\tilde{T}_{i,j}$; Then encrypt the keyword wi using HMAC, $\tilde{w}_i = HMAC(w_i, key)$ where key is the key of HMAC. At this point, the encrypted keyword - search tree binary $(\tilde{w}_i, \tilde{T}_{i,j})$ can be obtained. When all the keyword search tree binary sequences in $\mathcal{L}'$ are encrypted, the encrypted keyword search tree binary sequence $\mathcal{L}'$ is obtained;

(5.2) Sort the binary sequence of encryption keywords search tree in $\mathcal{L}'$ based on the size relationship of encryption keywords, and generate the binary sequence $\tilde{\mathcal{L}}$ of encryption keywords search tree, This sequence is the encrypted index.

Furthermore, in step (5.1), the AVL search tree $T_{i,j}$ is encrypted using 0-1 encoding and HMAC. The coordinate data c stored in each node of $T_{i,j}$ needs to be encrypted according to the following steps:

(5.1.1) For the coordinate data c stored in each node of $T_{i,j}$, first convert the coordinate data c into a binary form cb of w bits, c_b, c_b=$b_1b_2…b_{w-1}b_w$ (b_i∈{0,1}, i∈{1, 2, …, w}), and then encode c_b 1 to generate the 1 encoding set OC = { $b_1b_2…b_i$ | b_i = 1, 1⩽i⩽w};

(5.1.2) For each encoding x in the 1 encoding set OC, use HMAC encryption processing to generate HMAC encoding $HMAC(x, key)$, where key is the key of HMAC, and finally obtain the encrypted 1 encoding set $\widetilde{OC}$, When all nodes of Ti and j are encrypted, the encrypted AVL search tree $\tilde{T}_{i,j}$ is obtained.

V. Implementation of a Privacy Protection Space Keyword Range Query System

The privacy preserving spatial keyword range query based on KLI index designed in this article mainly includes four parts: document preprocessing, keyword search tree building, location point set tree building, and retrieval part. The main operational process is shown in Figure 5:

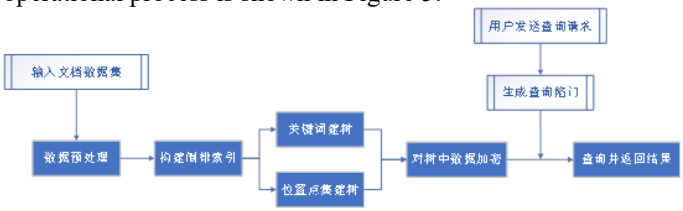

Fig. 5. Query process

A. Dataset preprocessing

This section mainly calls dataset.py to handle the content in the Yelp dataset. Yelp data is in JSON format, so it needs to be preprocessed according to our requirements. Firstly, extract all the keywords and generate a keyword set.

After generating the keyword set, KLI index can be generated. Based on the generated keyword set, search for merchants containing the keyword in the Yelp dataset's merchant comments, and save the merchant name to a dictionary. Finally, the dictionary will be serialized and saved to the file *kw_dict* for future use.

Finally, after converting the longitude and latitude coordinates of each position point into a geodetic coordinate system, a dictionary data is generated for each position and its converted position coordinates_ Store after loc. The pyproj library used for projection in this article is used to achieve various map projections and coordinate transformations.

In the Reverse Mercator projection, the Earth is viewed as an ellipsoid rather than a sphere. The basic idea of projection is to map all points on the surface of an ellipsoid onto a plane, while keeping the proportion of distance and the size of angle

unchanged. In this process, the meridians and parallels are projected as straight lines, while the curves are projected as curves. Due to the fact that the Reverse Mercator projection is a regular cone projection, it can maintain the size and shape of the angle, but it will change the proportion of distance.

B. Keyword tree building

This section calls kw_ Implement using a tree file. The key part lies in tree building and searching. During the process of building a tree, perform an OR operation on two nodes and use them as the parent node. Ultimately, all nodes will be merged into one tree.

Specifically, the input to this function is a list of nodes, where each node contains a data item and two pointers to left and right child nodes. During the process of creating a tree, if the length of the node list is odd, the last node in the list is copied and added to the end of the list for pairwise merging. Next, the function groups adjacent nodes in order and merges them into a new node. The data items of the new node are obtained through OR operation.

After completing a layer of node merging, the function recursively calls itself with the new node list as a parameter, continuing to merge nodes in the next layer until all nodes are merged into one tree. Finally, if the number of nodes in the current layer is 1, it indicates that it has been merged into the root node, and the node is directly returned as the root node of the tree.

The search section consists of two steps. The first step is to prune the query trapdoor, and the second step is to traverse the pruned tree to obtain the result. The pruning operation is implemented recursively. For each node, if the node is empty or the product of the node data and the trapdoor matrix is 0, the node and its subtree are removed from the tree. When recursively processing left and right subtrees, the same pruning operation is also used. If the node is a leaf node and the product of the node data and token matrix is 0, then the node is removed from the tree. Finally, if the node has not been deleted, it is returned; otherwise, it is returned as NULL. In search operations, starting from the root node, traverse the pruned tree and use a queue to store the nodes to be processed. When processing each node, if the node is not a leaf node, let it enter the queue to traverse its subtree. If the node is a leaf node, add it to the result list. The final result list is returned.

C. Location point tree building

This section establishes an AVL tree for the location point set by calling AVLTree. The main part is to briefly describe the insertion of data during the process. Firstly, based on the size relationship of keywords, insert a new node into the left or right subtree of that node. Then, the function checks whether the balance factor of the node exceeds the threshold (i.e. the height difference between the left and right subtrees is greater than 1). If it exceeds the threshold, a rotation operation is performed to adjust the structure of the tree to maintain its balance. Specifically, if the height of the left subtree is greater than that of the right subtree, and a new node is inserted into the left subtree, there may be an imbalance of LL or LR type. If a new node is inserted into the right subtree, there may be an imbalance of RR or RL type.

D. Implementation of Range Query

The entire query process is implemented by calling KLI.py. When the data owner receives a query request, they first convert it into a search trapdoor, replace the longitude and latitude coordinates of the current position with a geodetic coordinate system, calculate the range in the geodetic coordinate system based on the search range, and convert the query keywords into bit vectors.

When querying, the first step is to use KLI index to filter out the set of location points based on the keyword set of the query. Then, further filtering is performed on each leaf node, and results that meet the criteria are filtered based on the search range of longitude and latitude. Specifically, for each leaf node, call search_ The range function performs a range search on its data items to find all data items that meet the specified range of longitude or latitude. Then, take the intersection of the indexes of all data items that meet the conditions to obtain the final result set.

VI. CONCLUSION

This graduation project is based on the research of spatial keyword range query in cloud environment, and proposes a privacy preserving spatial keyword range query based on KLIAT index. Firstly, by combining the inverted index with the AVL tree, a keyword position inverted AVL tree index, namely KLIAT index, is generated; Then, in order to improve the query speed of keywords, the keywords are transformed into bit vectors and a keyword search tree is constructed from bottom to top. In order to achieve privacy protection range query, the location point coordinates were encrypted using 0-1 encoding and HMAC, and the keywords were encrypted using ASPE to construct a secure index. In order to verify the feasibility of the model, this paper also compared this scheme with the LSKQ scheme, verifying that the scheme has a faster query rate. Finally, a prototype system for privacy preserving spatial keyword range query was designed and implemented.

VI. REFERENCES

[1] Li R, Liu A X, Wang A L, et al. Fast and scalable range query processing with strong privacy protection for cloud computing[J]. IEEE/ACM Transactions On Networking, 2015, 24(4): 2305-2318.

[2] Xiao G, Wu F, Zhou X, et al. Probabilistic top-k range query processing for uncertain databases[J]. Journal of Intelligent & Fuzzy Systems, 2016, 31(2): 1109-1120.

[3] Xue K, Li S, Hong J, et al. Two-cloud secure database for numeric-related SQL range queries with privacy preserving[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(7): 1596-1608.

[4] Zhang C, Zhu L, Xu C, et al. Location privacy-preserving task recommendation with geometric range query in mobile crowdsensing[J]. IEEE Transactions on Mobile Computing, 2021, 21(12): 4410-4425.

[5] Song F, Qin Z, Liu D, et al. Privacy-preserving task matching with threshold similarity search via vehicular crowdsourcing[J]. IEEE Transactions on Vehicular Technology, 2021, 70(7): 7161-7175.

[6] Xie H, Guo Y, Jia X. A privacy-preserving online ride-hailing system without involving a third trusted server[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 3068-3081.

[7] Zheng Y, Lu R, Guan Y, et al. Achieving efficient and privacy-preserving exact set similarity search over encrypted data[J]. IEEE Transactions on Dependable and Secure Computing, 2020, 19(2): 1090-1103.

[8] Tong Q, Miao Y, Li H, et al. Privacy-preserving ranked spatial keyword query in mobile cloud-assisted fog computing[J]. IEEE Transactions on Mobile Computing, 2021.

[9] Choi S, Ghinita G, Lim H S, et al. Secure knn query processing in untrusted cloud environments[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(11): 2818-2831.

[10] Wang P, Ravishankar C V. Secure and efficient range queries on outsourced databases using Rp-trees[C]//2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, 2013: 314-325.

[11] Zhu H, Lu R, Huang C, et al. An efficient privacy-preserving location-based services query scheme in outsourced cloud[J]. IEEE Transactions on Vehicular Technology, 2015, 65(9): 7729-7739.

[12] Xu G, Li H, Dai Y, et al. Enabling efficient and geometric range query with access control over encrypted spatial data[J]. IEEE Transactions on Information Forensics and Security, 2018, 14(4): 870-885.

[13] Su S, Teng Y, Cheng X, et al. Privacy-preserving top-k spatial keyword queries in untrusted cloud environments[J]. IEEE Transactions on Services Computing, 2015, 11(5): 796-809.

[14] Meng X, Zhu H, Kollios G. Top-k query processing on encrypted databases with strong security guarantees[C]//2018 IEEE 34th International Conference on Data Engineering (ICDE). IEEE, 2018: 353-364.