Accurate Range Query with Privacy Preservation for Outsourced Location-Based Service in IoT

Liu, Zhaoman; Wu, Lei; Meng, Weizhi; Wang, Hao; Wang, Wei

Published in:
IEEE Internet of Things Journal

Link to article, DOI:
10.1109/JIOT.2021.3068566

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Citation (APA):
Accurate Range Query with Privacy Preservation for Outsourced Location-Based Service in IoT

Zhaoman Liu, Lei Wu, Weizhi Meng, Senior Member, IEEE, Hao Wang and Wei Wang

Abstract—With the maturity of Internet of Things technology, location-based service (LBS) is developing rapidly in intelligent terminal devices, and it brings new vitality to the fields of logistics, transportation, product traceability and so on. The popularity of LBS produces a lot of spatial data, which inevitably brings burden to the storage and management of LBS provider (LBSP). With the help of cloud computing and cloud storage, outsourcing spatial data to cloud server has become a new trend. However, due to the cloud server is not trusted, data outsourcing will face the problems of data disclosure and query disclosure. Range query is a common query in LBS, considering the situation of data outsourcing, this paper proposes an accurate range query (ARQ) scheme, which can realize efficient range query while preserving LBSP’s data privacy and user’s query privacy from being disclosed to the cloud server. The ARQ scheme is suitable for spatial data in any form without being limited to the case that the data points are only integers, which has a certain practical significance. In addition, by dividing the region into atomic regions, ARQ can realize sub-linear search time and ensure dynamic update of spatial data. We proved the security of the proposed scheme through security analysis, and demonstrated the effectiveness of the scheme through experiments.

Index Terms—IoT, LBS, Range query, Privacy preservation, Data outsourcing, Hilbert curve, SSW.

I. INTRODUCTION

With the continuous development of the Internet of Things, location-based services (LBS) also usher in huge development opportunities. The emergence of low-power wide-area network (LPWAN) makes it possible for wearable devices and urban infrastructure to access the network, and LBS will follow the access of devices to cover the whole network. A growing number of mobile devices have precise GPS positioning function, which makes LBS increasingly popular and becomes one of the most promising services in the minds of mobile users. The successful operation of LBS relies on a large amount of spatial data, which is not only used in LBS, but also widely used in computational geometry, medical imaging, earth science, etc. Although LBS brings convenience to the user’s life, it also poses a threat to the user’s personal safety and property safety. It is well known that the user needs to provide their location when enjoying location-based services, however, their location may involve other sensitive information, such as home address, living habits, health status and social relations. Therefore, how to obtain location-based services while protecting the user’s location privacy has attracted extensive attention from the society.

Along with the development of cloud storage and cloud computing technology, more and more location-based service companies tend to store spatial databases on the cloud. For example, Yelp and Foursquare rely on public clouds (such as Amazon Web Services, AWS) to manage their spatial data sets and process queries, which makes the interaction between users and cloud server replace the interaction between users and LBSP, thus reducing the storage and management burden of LBS companies. Although cloud services bring convenience to LBS companies, outsourcing data may cause data leakage since cloud servers are generally honest but curious. At the same time, as user directly sends their query requests to cloud server, it will inevitably lead to the disclosure of user’s query privacy. Data privacy and query privacy can be preserved by encrypting data sets and queries, but this makes it difficult for cloud server to search and match. Therefore, under the premise of preserving LBSP’s data privacy and the user’s query privacy, how to make the cloud server provide effective service for the user has become an urgent problem to be solved.

Range query is the basic query in LBS. For example, in the application of peripheral recommendation, when the user queries LBSP for banks within 1 km, the service provider will provide the user with near to far candidates according to the distance between the user and surrounding banks. In the interaction between the user and LBSP, if the user’s location and query content are known by lawbreakers, it will pose a threat to the user’s personal safety and property safety. In a range query, the position is viewed as a data point in Euclidean space, and the query is described as a geometric object, such as a circle, rectangle, or arbitrary polygon. Among them, the purpose of circular range query is to find the data points located in the circular range, which has been widely used in geographic information system, computational geometry and computational aided design.

Searchable encryption (SE) schemes can assist in the search
of outsourced encrypted data, meaning that the cloud server can search without knowing the data or the query content. Most existing SE schemes are suitable for keyword query but not for range query based on spatial data. Unlike keyword query that requires equality testing, range query based on spatial data requires compute-compare operations, such as circular range query needs to calculate the distance from the data point to the center of the circle and then compare the distance to the radius. The existing cryptographic primitives are inefficient in distance computation and comparison of ciphertext, which makes the design of SE schemes that support range queries more challenging. More specifically, Pseudo-Random Function (PRF) \cite{1} can only support equality testing; Order-Preserving Encryption (OPE) \cite{2} only supports comparison; Partial Homomorphic Encryption (PHE) \cite{3} can only calculate addition (such as Paillier encryption \cite{4}) or multiplication (such as Elgamal encryption \cite{4}). In principle, Full Homomorphic Encryption (FHE) \cite{5} can calculate addition and multiplication on ciphertext, but its efficiency is relatively low, and the calculation of encrypted data with FHE cannot directly expose the search results, that is, it cannot directly make clear that the data points are inside or outside the query range, which limits its use in search.

In this paper, we propose a range query scheme (ARQ). According to the user’s query range, it can accurately retrieve the encrypted spatial data without disclosing LSBP’s data privacy and the user’s query privacy to the cloud server. Compared with our previous scheme \cite{6}, this scheme can realize accurate range query. Compared with scheme \cite{7}, this scheme requires less storage space when constructing data points.

The main work and contributions of this paper are as follows:

1. We transform spatial data and query range into location vectors and a set of query vectors respectively. Instead of performing "calculate-compare" operations on encrypted spatial data and query range, the symmetric SSW algorithm \cite{8} is adopted to encrypt the location vector and query vector, and further judge the relationship between spatial data and query range confidentially, which avoids the complex calculations in homomorphic encryption.

2. We divide the region into atomic regions by Hilbert curve, and index the spatial data within the same atomic region, which enables our scheme to achieve sublinear search time and is suitable for query on large-scale datasets.

3. We design a vector construction method, in which the length of the vector is related to the atomic region’s edge length, and the value of the vector is related to the spatial data’s relative position in the atomic region. This construction enables our scheme to support query on arbitrary data without limiting to integers, which is more in line with the practical application scenario. In addition, the idea of transforming the query range into a vector set enables us to extend the query range to any shape, not just a circular range.

4. We formalize the definition of the leakage function of the scheme, and rigorously prove the data privacy and query privacy with indistinguishability under selective chosen plaintext attacks (IND-CPA). We evaluate ARQ and demonstrate that ARQ is efficient on real-world spatial data sets.

The rest of the paper is organized as follows. Section 2 introduces the relevant work on privacy preservation in range query and nearest neighbour query. Section 3 introduces the background knowledge used in this paper. Section 4 describes the system model and data storage model, as well as the query model and potential threat model of the system in detail. Section 5 describes the scheme of this paper. Section 6 and Section 7 respectively analyse the security and efficiency of the scheme from the theoretical and experimental perspectives. Finally, Section 8 summarizes the whole scheme and prospects the future work.

II. RELATED WORK

In this part, we summarize the privacy preservation schemes for range query and nearest neighbour query in LBS, and make a detailed comparison between the proposed scheme in this paper and the previous schemes for range query from the perspectives of cryptography primitives, security and efficiency, which is shown in Table I.

A. Range query with privacy preservation

At present, Some related works have studied the range query of encrypted data. According to the different query shapes, they can be roughly divided into rectangular, circular and arbitrary shape range query.

Rectangle range query needs to retrieve all data points within the rectangle range. The multi-dimensional range searchable encryption schemes \cite{9, 10} essentially provide a solution that supports the search of rectangular range. Specifically, Boneh et al. \cite{9} and Shi et al. \cite{11} designed a public key scheme that can process rectangular range queries with linear search time. By using the tree structure, such as R-tree \cite{12, 13} or kd-tree \cite{14}, the schemes \cite{10, 12–14} can perform the rectangular range query in a faster time than the linear search. However, these solutions do not support circular range queries on encrypted spatial data.

Circular range query retrieves all data points within the circular range. By using a set of concentric circles, Wang et al. \cite{15} proposed a scheme that could retrieve data points within the circular range from the encrypted data. Zhu et al. \cite{16} also established a circular range query scheme for encrypted spatial data. However, these two schemes only apply to the circular range and cannot be applied to other geometric shapes.

Wang et al. \cite{17} proposed the first work on query of generalized geometric range. The main idea of the scheme is to transform various geometric queries into the same form: a set of query points. Then they proposed a generalized geometric range query based on Bloom Filter \cite{18} and predicate encryption \cite{8}. The scheme in \cite{7} improves the search complexity by designing a new form of equal-vector \cite{17}. However, these scenarios require enumerating all the data points in a given geometry object and using time-consuming predicate encryption as the building block. Therefore, these schemes are not suitable for processing large-scale dataset. Luo et al. \cite{19} proposed a generalized geometric range query scheme for encrypted datasets based on ASPE \cite{20} and geometric
transformation [21] by replacing the geometric range with its circumscribed polygon. Compared with the existing literature [7, 10, 17], their scheme can conduct generalized geometric range query on the encrypted dataset and achieve practical efficiency at the same time. In addition, their scheme claims to be able to resist attacks under a known background model. Li et al. [22] studied the security of Luo’s scheme [19], proposed an attack method and showed that Luo’s scheme could be quickly destroyed under the known background model. At the same time, Li et al. [22] proposed a new geometric range search scheme, which can solve the security defects of Luo’s scheme [19] and is secure under the known background model. In addition, they improved the efficiency of geometric range search to the sub-linear order by using the $R$-tree index structure.

B. Nearest neighbor or $K$ nearest neighbor query with privacy preservation

Nearest neighbor (NN) or $K$ nearest neighbor (KNN) query refers to the retrieval of one or $k$ data points nearest to the query point according to Euclidean distance. [20] first studied the NN query or KNN query for encrypted data, which can realize the search in linear time but is vulnerable to the attack of the chosen plaintext. The recent schemes [23–26] improved Wong’s scheme [20], making the nearest neighbor query on the encrypted data more secure and efficient. Subsequent researches then focused on improving query efficiency by using data structures [27] or minimizing client-server interactions [28]. Scheme [29, 30] enables the user to receive the most relevant results from the encrypted dataset according to the predefined correlation scoring function; Yang et al. [31] further proposed a secure ranking scheme using Paillier [3] encryption, which can support multiple users and any language; The scheme proposed by Akavia et al. [32] can use full homomorphic encryption to retrieve the first matching record.

C. Spatial data queries between two parties

It is necessary to consider secure interactions between users or between user and LBSP because the third-party servers (such as the cloud server) are often not trusted in real life. Schemes [33–35] realized the calculation and comparison of the distance between the two parties (Alice and Bob) by using the secure two-parties computation. Among them, Alice has the confidential spatial data and Bob has the confidential query range. Alice and Bob can use Secure Multiple-party Computation (SMC) to determine whether the spatial data is within the query range or not without disclosing their privacy, which inevitably introduces multiple rounds of interaction between the two parties. In addition, using Garble Circuit [36] to perform addition, multiplication and comparison on ciphertexts also makes range query possible, but like homomorphic encryption, Garble Circuit will cause higher computational and communication complexity.

D. Location privacy preservation in Internet of things

Although location privacy preservation in the Internet of things has been studied in recent years, there are still many problems to be solved. First of all, researchers have proposed a lot of general privacy preservation mechanisms, but on the whole, less consideration is given to the actual application scenarios, and the practicability is poor. Among them, the more mature application scenario is the Internet of vehicles. Literatures [37, 38] studied the privacy leakage of vehicles and users in ride matching, and proposed a shared ride matching scheme to protect location privacy.

In addition, through big data and other related technologies, attackers can obtain the user’s location from a variety of channels, and infer the user’s privacy through data mining and other means. In this context, to encrypt database has become a very promising direction [39, 40], which provides data confidentiality and performs queries on encrypted data without sacrificing functions.

III. PRELIMINARIES

This section provides a brief overview of the encryption techniques used in the proposed solution.

A. Hilbert Curve

The space filling curve can map unordered data in the high-dimensional space to the one-dimensional space, through which the adjacent objects in the space will be stored in the adjacent one-dimensional space, which can not only reduce the time of input and output, but also improve the efficiency of data processing in memory. According to the characteristics of filling curve, the Hilbert curve can run through each discrete unit of two-dimensional space or higher dimensional space linearly once and only once, and conduct linear ordering and coding for each discrete unit, which serves as the unique identifier for the unit. The Hilbert curve specifies the order of points in a two-dimensional plane. At the root level, once a direction and a starting point are selected, the order of quadrants can be determined by surrounding four quadrants and numbering them from 0 to 3. When we want to determine the order of access to the sub-quadrants and maintain the overall adjacency property, we need to perform a simple transformation to the original curve. For a given quadrant, the curve we draw in it is determined by the curve above it and the position of that quadrant. As shown in Fig. 1, four transformations of the current layer can be determined according to the direction of the curve of the previous layer.

B. Shen-Shi-Waters Encryption

Shen, Shi and Waters (SSW) designed a symmetric key predicate encryption scheme that supports the inner product query, which can calculate whether the inner product of two
vectors is 0 without disclosing the privacy. Specifically, given two vectors \( \vec{u} = (u_1, u_2, \ldots, u_\alpha) \) and \( \vec{v} = (v_1, v_2, \ldots, v_\alpha) \), SSW generates ciphertext \([\vec{u}]\) of vector \( \vec{u} \) and ciphertext \([\vec{v}]\) of vector \( \vec{v} \). On the premise of not disclosing \( \vec{u} \) and \( \vec{v} \), the calculation of \([\vec{u}]\) and \([\vec{v}]\) indicates whether the inner product of \( \vec{u} \) and \( \vec{v} \) is 0, namely

\[
\begin{cases}
    \text{if } \langle \vec{u}, \vec{v} \rangle = 0, & \text{SSW.Query([\vec{u}], [\vec{v}]) = 1} \\
    \text{otherwise,} & \text{Pr}[\text{SSW.Query([\vec{u}], [\vec{v}]) = 0}] \geq 1 - \text{negl}(\lambda),
\end{cases}
\]

where \( \langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{\alpha} u_i \cdot v_i \) is the inner product of two vectors. In addition to protecting data privacy, SSW can also protect query privacy. The security of SSW is proved to be indistinguishable under selective chosen-plaintext attack. Detailed security analysis of SSW can be found in [8]. The algorithm of SSW is briefly introduced as follows:

Setup(\( \lambda \)): Given the security parameter \( \lambda \), output the secret key \( sk \);

Enc(\( sk, \vec{u} \)): Given the secret key \( sk \) and the vector \( \vec{u} \), where \( \vec{u} = (u_1, u_2, \ldots, u_\alpha) \), output the ciphertext \([\vec{u}]\);

GenToken(\( sk, \vec{v} \)): Given the secret key \( sk \) and the vector \( \vec{v} \), where \( \vec{v} = (v_1, v_2, \ldots, v_\alpha) \), output the token \([\vec{v}]\);

Query(\( [\vec{u}], [\vec{v}] \)): Given the ciphertext \([\vec{u}]\) and the token \([\vec{v}]\), if \( \langle \vec{u}, \vec{v} \rangle = 0 \), output 1 and output 0 otherwise.

The encryption time and token generation time of SSW are both \( \mathcal{O}(\alpha) \), and the size of ciphertext and token are both \( \mathcal{O}(\alpha) \), where \( \alpha \) is the vector length.

### IV. SYSTEM AND THREAT MODELS

#### A. System Model

As shown in Fig. 2, the system is composed of user, LBSP and cloud server provider (CSP). The user generates encrypted request of range query and sends it to CSP; LBSP stores location information of intelligent terminal devices, processes spatial dataset and sends the encrypted data information to the CSP; CSP retrieves the encrypted dataset according to the user’s encrypted request, re-encrypts the spatial data that satisfies the query request, and returns the re-encrypted dataset to user, finally the user decrypts and obtains spatial data within the query range.

The system can be divided into initialization phase and query processing phase. In the initialization phase, LBSP uses the Hilbert curve to further divide the region into atomic regions and generates the key required for the query phase. When a user registers with LBSP, LBSP generates public and private keys for encrypting and decrypting the regional partition mode and generates key for re-encrypting spatial data, and sends the above keys with the key for encrypting the query range together to the user via a secure channel. In the query processing phase, LBSP constructs vector for spatial data and builds index for the spatial dataset within the same atomic region. After that, LBSP sends encrypted indexes to CSP. Once the user generates a query request, it firstly requests the atomic region coding of the region to LBSP, then LBSP encrypts the regional partition mode with the user’s public key and sends the encrypted mode to the user. After decryption, the user selects atomic regions that need to query and constructs a set of query vectors, both of which represent the query range. The query range is encrypted and sent to CSP together with re-encrypted key. CSP matches the encrypted query with the encrypted index and re-encrypts the spatial data that meets the query range to the user. At last, the user decrypts and obtains the spatial data within the circular range.

#### B. Storage Model

1) Region division: This paper considers dividing the region on the basis of "country-province-city-region". Under such circumstance, LBSP uses Hilbert curve to further divide the region into atomic regions, which are viewed as the basic unit to store and manage spatial data. LBSP presets the threshold \( d \) so that the side length of the divided atomic region is not greater than \( d \). Due to the monotonicity of Hilbert

#### TABLE I

<table>
<thead>
<tr>
<th>Range</th>
<th>Scheme</th>
<th>Methods</th>
<th>Security</th>
<th>Efficiency</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>rectangular</td>
<td>[10] Asymmetric Hidden Vector Encryption</td>
<td>Ensure single dimensional privacy and selective security</td>
<td>Faster than linear</td>
<td>Query privacy is not guaranteed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[12] Asymmetric Scalar-Product Preserving Encryption</td>
<td>Resist known-plaintext attack and ordering information leakage</td>
<td>Faster than linear</td>
<td>Lack of formal security definition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[13] Symmetric-key SSW</td>
<td>Ensure single dimensional privacy and selective security</td>
<td>Faster than linear</td>
<td>Cannot extend to range queries against other shapes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[14] Symmetric-key range predicate encryption</td>
<td>Resist chosen-plaintext attack</td>
<td>Logarithmic-time</td>
<td>Single-dimensional privacy is not guaranteed</td>
<td></td>
</tr>
<tr>
<td>circle</td>
<td>[15] Symmetric-key SSW</td>
<td>Resist chosen-plaintext attack</td>
<td>Linearly related to the query radius</td>
<td>Spatial data and query radius are limited to integers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[16] Improved homomorphic encryption</td>
<td>Ensure data privacy and query privacy</td>
<td>Faster than linear</td>
<td>Cannot extend to range queries against other shapes</td>
<td></td>
</tr>
<tr>
<td>Any shape</td>
<td>[17] Symmetric-key SSW</td>
<td>Ensure data privacy and query privacy</td>
<td>Sublinear and related to dataset size</td>
<td>Cannot apply to large scale dataset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[19] Asymmetric Scalar-Product Preserving Encryption</td>
<td>Resist attacks under a known background model</td>
<td>Sublinear</td>
<td>The matching result is not accurate enough</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>Symmetric-key SSW</td>
<td>Ensure data privacy and query privacy</td>
<td>Sublinear</td>
<td>Cannot resist collusion attacks</td>
<td></td>
</tr>
</tbody>
</table>

#### TABLE I COMPARISON OF SCHEMES ON RANGE QUERY WITH PRIVACY PRESERVATION

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3068566, IEEE Internet of Things Journal
for every atomic region in \( CR_2 \). Among them, \( CR_1 \) refers to the atomic regions that are in the query range, \( CR_2 \) refers to the atomic regions that are intersecting with the query range, and \( \{ \bar{v} \} \) represents the data points in the form of vector which are in \( CR_2 \) and query range simultaneously. The user requires cloud server to return all the spatial data in \( CR_1 \) and return the spatial data whose query result is 1 in \( CR_2 \). In order to protect user’s query privacy, it is necessary to encrypt the atomic region sets and query vectors before the user sends a request to CSP. As shown in Fig. 5, in the scenario of using the third-order Hilbert curve to divide the region, the user requests range query at point \( O \), and the query radius is expressed by \( r \). In this case \( CR_1 \)=\{10, 11, 28, 29, 30, 31, 32\}, \( CR_2 \)=\{8, 9, 12, 17, 18, 24, 27, 33, 34, 35, 36, 53, 54\}.

![System model for outsourced range query](image)

**Fig. 2.** System model for outsourced range query

because it is difficult to reveal the specific region that the coding represents without knowing the generating rule of curve. The region division by Hilbert curve with different orders is shown in Fig. 3.

![Region division by Hilbert Curve](image)

**Fig. 3.** Region division by Hilbert Curve with different order

2) Storage structure: For any spatial data, it consists of the following parts: (1) the region name \( R \), (2) the atomic region coding \( AR \), (3) data number \( N \), (4) location vector \( \bar{v} \) and (5) spatial data \( D \). The region name refers to the area where the spatial data is located. Location vector represents the coordinate information of spatial data in the form of vector, which is used to compare with the query vector. Due to the privacy of spatial data, LBSP processes atomic region coding, location vector and spatial data by encryption. The storage structure of spatial data is shown in Fig. 4. After transforming spatial data, LBSP will logically connect spatial data distributed in the same atomic region to form a linked list.

![Storage structure of spatial data](image)

**Fig. 4.** The storage structure of spatial data

### C. Query Model

The query request generated by user includes the following parts: (1) the region name \( R \), (2) a set of atomic regions \( CR_1 \), (3) a set of atomic regions \( CR_2 \) and (4) query vector set \( \{ \bar{v} \} \)
matching operations together with the query vectors sent by user. Based on these operations, CSP may try to link the query range with the retrieved data.

V. A CONCRETE CONSTRUCTION OF ACCURATE RANGE QUERY SCHEME

The scheme designed in this paper includes LBSP, CSP and user. The whole scheme can be divided into the following phases: (1) initialization phase, (2) index generation phase, (3) database encryption phase, (4) query generation phase, (5) token generation phase, (6) matching and re-encryption phase, and (7) decryption phase. Each phase is described in detail next.

A. Initialization

This phase is performed by LBSP. LBSP uses Hilbert curve to divide the region into atomic regions, each of which is encoded by a unique identifier. Select a random number \( k_1 \) as the key of hash function \( H \) to encrypt the Hilbert curve coding. At the same time, LBSP calculates the symmetric key \( k \), which is used to encrypt the location vector and query vectors. Let \( \mathcal{G} \) be a group generator algorithm. LBSP calls \( \mathcal{G}(1^k) \) to obtain \((p, q, r, s, G, G_T, \epsilon)\), where \( p, q, r, s \) are random prime number and \( G = G_p \times G_q \times G_r \times G_s \) is a \( N \)-order composite group \( (N = pqrs) \), and then LBSP selects generators \( g_p, g_q, g_r, g_s \) from group \( G_p, G_q, G_r, G_s \) respectively. For \( i = 1 \) to \( \omega \), LBSP selects \( h_{1,i}, h_{2,i}, u_{1,i}, u_{2,i} \in G_p \) uniformly and randomly, where \( \omega \) is the length of vector, and calculates symmetric key

\[
k = (g_p, g_q, g_r, g_s, (h_{1,i}, h_{2,i}, u_{1,i}, u_{2,i})_{i=1}^\omega).
\]

In addition, LBSP generates public key and private key \((pk_p, sk_p)\) for encrypting and decrypting spatial data.

When a user \( u_i \) registers with LBSP, LBSP generates identity \( id_i \) and key pair \((pk_i, sk_i)\) for the user, in which the key pair is used to secretly interact the regional division mode with LBSP. At the same time, LBSP uses its private key \( sk_p \) and user’s public key \( pk_i \) to generate the re-encryption key \( rk_{p,i} \), which is used to re-encrypt the spatial data. Finally, LBSP sends \( param = \{id_i, pk_i, sk_i, rk_{p,i}, sk\} \) to the user through the secure channel, where \( sk = \{k_1, k\} \).

B. Index Generation

This phase is performed by LBSP. Given a spatial dataset \( DB \), which includes the location information of intelligent terminal devices, LBSP needs to process the spatial data before outsourcing it to CSP. As shown in Fig. 4, each spatial data should contain information such as the region \( R \) that it belongs to, the atomic region \( AR \) that it belongs to, data number \( N \), location vector \( \vec{v} \) and data content \( D \), where location vector is used to determine whether the spatial data falls within the user’s query range, and data content \( D = (x, y) \) indicates the specific location of spatial data.

When constructing the location vector, LBSP needs to determine the length of the vector according to the actual application. Instead of the real position \((x, y)\) of data point \( D \), we use its relative position \((x_H, y_H)\) in the atomic region to represent the location vector, which obviously shortens the vector length and ensures the unification of the vector length in different atomic regions, thus avoiding the inference attack caused by the different vector length. Suppose the edge length of atomic region is an integer \( d \), and the data points keep \( l \) decimal places in each dimension, then the space size of each dimension in the atomic region is \( [0, 10^l d] \), and the x-value and y-value can be represented by \( \lfloor l \log_2 10 + \log_2 d \rfloor \) bits. Among them, \( S \) is the side length \( d \) of the atomic region is inversely proportional to the order \( N \) and directly proportional to the length of \( S \) of the region, which can be expressed as \( d = S/2^N \). We use \( \omega = 2[\log_2 10 + \log_2 d] + 1 \) bits to represent the location vector, and the last bit is the verification bit whose value is fixed to 1, which is used to check whether the location vector matches the query vector. For ease of understanding, Fig. 6 describes the distribution of data points with one decimal place in the atomic region with \( d = 1 \) and the form of location vector. According to our construction rule, data points need to be represented by 9 bits, and each dimension occupies 4 bits. The above construction method takes into account the fact that the value of data points in practical application is not completely integer. Through this construction method, our scheme can query any data points in a query range.

LBSP regards the atomic region as the basic unit of data storage and management. It links the spatial data distributed in the same atomic region into a linked list and considers it as an element of region \( R \), and finally the spatial data in region \( R \) will form an index \( \Gamma \) as shown in Fig. 7. The algorithm description of this stage is shown in algorithm 1, where \( m \) represents the number of atomic regions formed after the region \( R \) is divided, and \( |AR_i| \) represents the number of spatial data contained in the atomic region \( AR_i \).

C. Database Encryption

This phase is performed by LBSP. Since the location of terminal device is confidential information of LBSP, in order to protect data’s privacy, LBSP needs to further encrypt the index \( \Gamma \). Specifically, LBSP will encrypt the atomic region coding, location vector and data content in the index. Firstly, \( H_{AR} \leftarrow H(k_1, AR) \) is obtained by hashing the coding \( AR \) of the atomic region.

\[\begin{align*}
\Gamma & \leftarrow \text{null} \\
\{AR_1, AR_2, \ldots, AR_m\} & \leftarrow \text{Hilbert}(R) \\
\text{for } i & \leftarrow 1, 2, \ldots, m \\
\Gamma_i & \leftarrow \text{Linklist.Init}() \\
\Gamma & \leftarrow \text{Linklist.Append}((\Gamma_i, R, \{AR_i\}) \\
\text{for } j & \leftarrow 1, 2, \ldots, |AR_i| \\
\vec{v}_i & \leftarrow \{0, 1\}^\omega \\
\Gamma_{ij} & \leftarrow (j, \vec{v}_i, D_{ij}) \\
\Gamma_i & \leftarrow \text{Linklist.Append}((\Gamma_i, \Gamma_{ij}) \\
end for \\
\Gamma & \leftarrow \Gamma \cup \{\Gamma_1\} \\
end for \\
return \Gamma
\end{align*}\]
region with key \( k_1 \). Then, the encrypted location vector \( [\vec{v}] \) is calculated. For location vector \( \vec{v} \in \{0, 1\}^\omega \), LBSP randomly selects \( a, b, \alpha, \beta \in \mathbb{Z}_N \), \( S_1, S_2 \in \mathbb{G}_a \), \( R_{1,i}, R_{2,i} \in \mathbb{G}_r \) for \( i = 1, \cdots, \omega \), and calculates the encrypted location vector

\[
[\vec{v}] = (C, C_0, \{C_{1,i}, C_{2,i}\}_{i=1}^\omega),
\]

where \( C = S_1 \cdot g_1^a \cdot g_0, C_0 = S_2 \cdot g_2^b \), and for \( i = 1, \cdots, \omega \),

\[
C_{1,i} = h_{1,i}^a \cdot u_{1,i}^b \cdot g_{1,i}^\alpha \cdot R_{1,i}, \quad C_{2,i} = h_{2,i}^a \cdot u_{2,i}^b \cdot g_{2,i}^\beta \cdot R_{2,i}.
\]

In addition, the encrypted data content \( C_D \) are calculated, where \( C_D \leftarrow Enc(pk_D, D) \).

Eventually, LBSP sends the encrypted index \( \Gamma^* = (R, H_{AR}, N, [\vec{v}], C_D) \) to the CSP. An algorithm for this phase is shown in Algorithm 2.

**Algorithm 2 DBEnc**(\( sk, \Gamma \))\( \rightarrow \Gamma^* \)

1: \( \Gamma^* \leftarrow \text{null} \)
2: extract \( k_1, k \) from \( sk \)
3: for \( i \leftarrow 1, 2, \cdots, n \) do
4: \( \text{extract } AR_i \text{ from } \Gamma_i \)
5: \( \forall \Gamma^*_i \leftarrow \text{Linklist.Init}() \)
6: \( H_i \leftarrow H(k_1, AR_i) \)
7: \( \forall \Gamma^*_i \leftarrow \text{Linklist.Append}(\Gamma^*_i, R, H_i) \)
8: for \( j \leftarrow 1, 2, \cdots, |AR_i| \) do
9: \( [\vec{v}_j] \leftarrow \text{SSW.Enc}(k, v_j) \)
10: \( C_{D,j} \leftarrow Enc(pk_D, D_j) \)
11: \( \forall \Gamma^*_j \leftarrow (j, [\vec{v}_j], C_{D,j}) \)
12: \( \forall \Gamma^* \leftarrow \text{Linklist.Append}(\Gamma^*_i, \Gamma^*_j) \)
13: end for
14: \( \Gamma*_i \leftarrow \Gamma^* \cup \{\Gamma^*_i\} \)
15: end for
16: return \( \Gamma^* \)

**D. Query Generation**

This phase is accomplished by user and LBSP. Before requesting range query, user \( u_i \) first asks LBSP for the division mode of region \( R \). By default, user and LBSP share the hierarchical pattern of “country-province-city-region”. After receiving the request from user \( u_i \), LBSP encrypts the division mode of region \( R \) with the user’s public key \( pk_i \) and sends it to the user. Finally, the user decrypts it by its private key \( sk_i \), and obtains the atomic region codes of region \( R \).

After obtaining the atomic region codes of region \( R \), the user determines the query range \( Q = (CR_1, CR_2) \) according to its location \( (x_u, y_u) \) and query radius \( r \). Among them, \( CR_1 \) represents the set of atomic regions within the query range, and \( CR_2 \) represents the set of atomic regions intersecting with the query range. For each atomic region in \( CR_2 \), a query vector set \( \{[\vec{u}]\} \) is constructed, which contains all the data points in the query range. Like the location vector, the query vector is represented by \( \omega = 2\lceil \log_2(10 + \log_2 d) \rceil + 1 \) bits, and the last bit is used to verify whether the query vector matches the location vector. In order to reduce the comparison with location vector, we use wildcard to merge multiple query vectors. Fig. 8 illustrates the construction rules of query vector. Specifically, set the wildcard bits to be 0, the 0-value in the non-wildcard bits to be -1, and the last bit to be -\( \omega_1 \), where \( \omega_1 \) represents the number of bits with the value of 1 in the initial vector. This construction rule fully considers the adverse effect that the product of 0 and any value is 0, and does not need to consider the value of location vector in the wildcard bits.

At last the user generates query \( Q = (R, CR) \). The algorithm description of this stage is shown in algorithm 3, where \( q \) represents the query range with query location \( (x_u, y_u) \) as the center of the circle and \( r \) as the query radius.

**E. Token Generation**

This phase is completed by the user. In order to preserve user’s query privacy, the query \( Q \) needs to be encrypted before it is sent to the CSP. Specifically, the atomic region codes involved in query \( Q \) and query vectors need to be encrypted. Firstly, hash the atomic region codes in set \( CR \) with the key \( k_1 \) to obtain \( H_{CR} = (H_{CR_1}, H_{CR_2}) \). Secondly, calculate the encrypted query vector \( [\vec{u}] \) with the key \( k \). For query vector \( \vec{u} \in \{-1, 0, 1\}^{\omega - 1} \) \( \setminus \omega \), the user randomly selects \( f_1, f_2 \in \mathbb{Z}_N, r_{1,i}, r_{2,i} \in \mathbb{Z}_N \) for \( i = 1, \cdots, \omega \), \( R_1, R_2 \in \mathbb{G}_r \),

<table>
<thead>
<tr>
<th>Integral Points ((x_i, y_i))</th>
<th>Location Vector ((s_i))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,1)</td>
<td>(001000011)</td>
</tr>
<tr>
<td>(3,6)</td>
<td>(001101101)</td>
</tr>
<tr>
<td>(4,4)</td>
<td>(010001001)</td>
</tr>
<tr>
<td>(7,2)</td>
<td>(011101001)</td>
</tr>
<tr>
<td>(8,8)</td>
<td>(100010001)</td>
</tr>
</tbody>
</table>
Algorithm 3 QueryGen(q)→Q
1: $CR_1 ← \text{null}$
2: $CR_2 ← \text{null}$
3: for $i ← 1, 2, \ldots, m$ do
4:  if $AR_i \in q$ then
5:   $CR_1 ← CR_1 \cup \{AR_i\}$
6:  else if $AR_i \cap q \neq \emptyset$ then
7:   select all $(x, y) \in AR_i \cap q$
8:   transform $(x, y)$ into $\vec{u}$
9:   $\vec{u} ← (-1, 0, 1)^{\omega-1} || - \omega_1$
10:  $CR_2 ← CR_2 \cup \{AR_i, \{\vec{u}\}\}$
11: end if
12: end for
13: $CR ← \{CR_1, CR_2\}$
14: $Q ← (R, CR)$
15: return $Q$

Finally, CSP re-encrypts the dataset $Res$ to get result set $Res'$ and sends $Res'$ to user, where $Res' ← \text{ReEnc}(r_{k_{p-1}}, Res)$. The algorithm description for this stage is shown in algorithm 5.

Algorithm 4 TokenGen(sk, Q)→TK
1: $H_{CR_1} ← \text{null}$
2: $H_{CR_2} ← \text{null}$
3: extract $CR$ from $Q$
4: for all $AR \in CR_1$ do
5:   $H_{CR_1} ← H_{CR_1} \cup \{H(k_1, AR)\}$
6: end for
7: for all $AR \in CR_2$ do
8:   $\{[\vec{u}]\} ← SSW.GenToken(k, \{\vec{u}\})$
9: $H_{CR_2} ← H_{CR_2} \cup \{H(k_1, AR), \{[\vec{u}]\}\}$
10: end for
11: $H_{CR} ← \{H_{CR_1}, H_{CR_2}\}$
12: $TK ← (R, H_{CR})$
13: return $TK$

Algorithm 5 Match($\Gamma^*$, TK)→Res
1: $Res ← \text{null}$
2: extract $\Gamma_1^*, \Gamma_2^*, \ldots, \Gamma_m^*$ from $\Gamma^*$
3: extract $H_{CR}$ from TK
4: for $i ← 1, 2, \ldots, m$ do
5:  extract $H_i$ from $\Gamma_i^*$
6:  if $H_i \in H_{CR}$ then
7:     $Flag=1$
8:  for $j ← 1, 2, \ldots, |H_i|$ do
9:    $Res ← Res \cup \{C_{D_j}\}$
10: end for
11: end if
12: end if
13: for $j ← 1, 2, \ldots, |H_i|$ do
14:  if $SSW.Query'([\vec{u}_i], [\vec{v}_j]) = 1$ then
15:     $Flag=1$
16:     $Res ← Res \cup \{C_{D_j}\}$
17: end if
18: end for
19: end if
20: end for
21: return $Res$

F. Matching and Re-encryption
This phase is performed by CSP. When CSP receives the query token $TK$ from the user, it adopts different query strategies for the two atomic region sets $H_{CR_1}$ and $H_{CR_2}$ in the token. First of all, CSP matches the set $H_{CR_1}$ with the atomic region codes in encrypted index $\Gamma^*$, and saves the encrypted spatial data from the matched atomic regions to the set $Res$. Then, CSP matches the set $H_{CR_2}$ with the atomic region codes in encrypted index $\Gamma_i^*$, with regard to the matched atomic regions, CSP computes $e(C, K) \cdot e(C_0, K_0) \cdot \prod_{i=1}^{m} e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}) \neq 1$ to judge the relationship between spatial data and query range and saves the encrypted spatial data whose judgment result is 1 to the set $Res$. 
G. Decryption

This phase is performed by the user. After receiving the re-encrypted result set $Res'$, the user needs to decrypt it with its private key $sk_i$, and finally obtains the location information of terminal devices within the query range. In this paper, we use proxy re-encryption technology [41] to realize the function of encrypting by LBSP and decrypting by users. In addition, Attribute-Based Encryption [42, 43] can also achieve fine-grained access control for spatial data with different attributes. This paper mainly focuses on how to achieve accurate range query for arbitrary form of spatial data. We will not elaborate too much on the proxy re-encryption of spatial data.

Correctness

For the atomic regions that intersects with the query range, CSP calculates $\bar{E} = e(C, K) \cdot e(C_0, K_0) \cdot \prod_{l=1}^{\omega} e(C_{1,l}, K_{1,l}) \cdot e(C_{2,l}, K_{2,l})$ to determine whether the encrypted spatial data in above atomic regions is within the query range. We can further deduce the above equation as follows:

$$E = e(C, K) \cdot e(C_0, K_0) \cdot \prod_{l=1}^{\omega} e(C_{1,l}, K_{1,l}) \cdot e(C_{2,l}, K_{2,l})$$

$$= e(S_1, g^a, R_1 \cdot \prod_{i=1}^{\omega} h^a_{1,i} \cdot h^a_{2,i} \cdot e(S_2, g^b, R_2) \cdot \prod_{i=1}^{\omega} h^b_{1,i} \cdot h^b_{2,i} \cdot e(S_3, g^c, R_3) \cdot \prod_{i=1}^{\omega} h^c_{1,i} \cdot h^c_{2,i})$$

$$= e(g^a, g^b, g^c) \cdot (\alpha f_1 + \beta f_2) v_{ui}$$

$$= e(g, g^a, (\alpha f_1 + \beta f_2) v_{ui}(u, v)$$

where $\langle u, v \rangle$ is the inner product of the vector $\vec{u}$ and $\vec{v}$.

According to the construction rules of location vector and query vector, when the data point D is in the query range, there exists a query vector in the atomic region where D is located, so that the inner product of the location vector $\vec{u}$ and the query vector $\vec{v}$ is 0, that is $\langle \vec{u}, \vec{v} \rangle = 0$. Specifically, for data point $D(x, y)$, the location vector formed by D is $\vec{v} = (b_1, \ldots, b_{\omega+1}, b_{\omega+2}, \ldots, b_{\omega-1}, b_0) \in \{0, 1\}^{\omega}$, where $\{b_1, \ldots, b_{\omega-1}\}$ represents the binary code of $xy$, $\{b_{\omega+1}, \ldots, b_{\omega-1}\}$ represents the binary code of $y_H$, and $b_0 = 1$. If D is in the query range, and the query vector corresponding to D is $\vec{u} = (b_1, \ldots, b_{\omega+1}, b_{\omega+2}, \ldots, b_{\omega-1}, b_0)$, where the wildcard bits generated by merging query vectors are set to be 0 and as for the non-wildcard bits from 1 to $\omega-1$, if $b_i = 0$, then $b_i = 1$, otherwise $b_i = b_i$. In addition, $b_0 = -1$, where $\omega$ is the number of bits with the value of 1 in $b_1, \ldots, b_{\omega-1}$, that is $\omega = \sum_{i=1}^{\omega-1} b_i$ for $b_i = 1$. Thus we can get $\langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{\omega} b_i b_i' = 0 -1 + \ldots + 0 \cdot (-1) + 1 \cdot 1 + \ldots + 0 -1 - \omega_1 = 0$.

When D is not in the query range, for any query vector in the atomic region where D is located, there exists at least one position i such that $b_i' = 1$ while $b_i = 0$, then $\langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{\omega} b_i b_i' = 0 -1 + \ldots + 0 \cdot (-1) + 0 -1 + 0 -1 + \ldots + 0 -1 - \omega_1 = 0$.

To sum up, when the spatial data $(x, y)$ is within the query range, $\langle u, v \rangle = 0$ and $E = 1$, then the judgment result of the Match algorithm is 1.

When the spatial data $(x, y)$ is on the boundary of the query range, $\langle u, v \rangle = 0$ and $E = 1$, then the judgment result of the Match algorithm is 1.

When the spatial data $(x, y)$ is outside the query range, $\langle u, v \rangle \neq 0$, and there exists two kinds of results:

- $E \neq 1$, then the judgment result of the Match algorithm is 0;
- $E = 1$ if and only if $\alpha f_1 + \beta f_2 v_{ij} = 0$, then the judgment result of the Match algorithm is 1 and $Pr\{E = 1\} \leq negl(\lambda)$.

In summary, we can get

$$\begin{cases} \text{if } \langle u, v \rangle = 0, & \text{Flag} = 1 \\ \text{else, } & Pr[\text{Flag} = 0] \geq 1 - negl(\lambda) \end{cases}$$

Therefore, according to the above algorithm we can judge the relationship between spatial data and query range: when the judgement result of the Match algorithm is 0, it means that spatial data is not in the query range; otherwise, the probability that spatial data is not in the query range is no more than a negligible function.

Accurate and extensible query. The ARQ scheme proposed in this paper transforms data point and query range into location vector and query vector respectively, and expresses the relationship between them through the inner product of vectors. This transformation enables the scheme to query arbitrary data points without limiting the value of spatial data to integers, which is more suitable for practical application. In addition, the method that transforms query range into query vector set enables our scheme to be extended to query spatial data within a range of arbitrary shapes. To sum up, our ARQ scheme can query arbitrary spatial data in any range.

Sublinear search. In ARQ scheme, the region R is divided into sets of atomic regions by using Hilbert curve, and the spatial data is stored and managed in atomic regions. When performing a query on a query range, the ARQ scheme first needs to find the atomic regions covered by the query range, and it requires $O(1)$ to find whether an atomic region matches the query range. For the atomic region in the query range, the spatial data contained in it can meet the query requirements, so no further analysis is needed; For the atomic region intersecting with the query range, we need to calculate whether each spatial data contained in it meets the query requirements by calling $SSWQuery$ algorithm, which requires $O(\alpha)$, where $\alpha$ is the average number of query vectors contained in the atomic region. Therefore, the total time required for a range query is $O(\alpha \tau)$, where $\tau$ represents the total amount of spatial data in the atomic region intersecting with the query range. However, it requires $O(n)$ to query all encrypted spatial data in region $R$, where $n$ is the total amount of spatial data in region $R$. Compared with $n$, it is obvious that $\tau$ is sublinear.

Effective update. The ARQ scheme can effectively update encrypted spatial data (including insert, delete or modify...
encrypted spatial data). This is because the ARQ scheme stores spatial data in the unit of atomic region and each encrypted spatial data is a separate linked list. Therefore, making changes to one encrypted spatial data does not require adjusting other spatial data. In ARQ scheme, it requires $O(\tau')$ to update an encrypted data point (i.e., $O(\tau')$ is required to find the data point to be updated, and $O(1)$ is required to update the data point), where $\tau'$ represents the number of data points in the same atomic region as the updated data point.

VI. SECURITY ANALYSIS

The ARQ scheme proposed in this paper uses Hilbert curve and SSW as building blocks, which not only achieves accurate range query, but also protects LBSP’s data privacy and user’s query privacy. In this part, we first define the leakage function, which represents all the information that can be captured by the adversary during the query. Then, we define data privacy and query privacy under the selective chosen-plaintext attack model. Finally, we prove that our scheme meets the security objectives.

A. Leakage Function

LBSP encrypts spatial data and sends them to CSP before the user requests a range query. When a user makes a query request to CSP, CSP retrieves the encrypted spatial dataset based on the user’s encrypted query and returns the spatial data that meets the query criteria to the user. To make the query processing run smoothly, CSP inevitably needs to know some information about spatial data and query requests. Specifically, given a spatial dataset and a sequence of range queries, in addition to public information such as security parameter $\lambda$, vector space $\Delta^\tau_p$, etc., information such as data size, data structure, retrieval results are also granted to untrusted cloud servers, which are formally described as leakage function $L$ and detailed as follows:

Definition 1. Size Pattern ($\varphi_1$): The amount of spatial data in region $R$. For Size pattern, the input of $L$ is an encrypted index $\Gamma^*$ of a spatial dataset, and the output is an integer $\varphi_1 = n \leftarrow L(\Gamma^*)$, where $n$ represents the number of data points within the region $R$.

Definition 2. Structure pattern ($\varphi_2$): The number of atomic regions containing data points and the number of spatial data contained in each atomic region. For Structure pattern, the input of $L$ is the encrypted index $\Gamma^*$ of the spatial dataset within a region $R$, and the output is a $m'$-dimensional vector $\varphi_2 = (|AR_1|, |AR_2|, \cdots, |AR_{m'}|) \leftarrow L(\Gamma^*)$, where $m'$ denotes the total number of atomic regions that contains data points in region $R$, and $|AR_i|$ denotes the size of the $i$th atomic region containing spatial data ($1 \leq i \leq m'$ and $\sum_{i=1}^{m'} |AR_i| = n$).

Definition 3. Query-size pattern ($\varphi_3$): The size of the range query, that is, the number of atomic regions involved in the range, which is divided into two parts: the first part is the number of atomic regions within the range, and the second part is the number of atomic regions intersecting the query range. For Query-size pattern, the token $TK$ is regarded as the input of $L$, where $TK \leftarrow TokenGen(sk, Q)$, $Q \leftarrow QueryGen(q)$, and the output of $L$ is a two-dimensional vector $\varphi_3 = (|CR_1|, |CR_2|) \leftarrow L(TK)$, where $|CR_1|$ denotes the number of atomic regions within the query range and $|CR_2|$ denotes the number of atomic regions on the boundary of the query range.

Definition 4. Search pattern ($\varphi_4$): The number of the same atomic regions in the current query compared to the previous queries. For Search pattern, the inputs of $L$ are token $TK$ and a previous token set $TK' = \{TK'_1, TK'_2, \cdots, TK'_t\}$, where $t$ represents the number of previous range queries, the output of $L$ is a $t$-dimensional vector $\varphi_4 = (\alpha_1, \alpha_2, \cdots, \alpha_t) \leftarrow L(TK, TK')$, where $\alpha_i$ represents the number of the same atomic regions compared current query $Q$ with the $i$th query $Q_i$. If $\alpha_i = 0$, then it means that there is no intersection between current query $Q$ and the $i$th query $Q_i$; and if $\alpha_i = |\varphi_3|$, it means that the current query $Q$ is a subset of the $i$th query $Q_i$.

Definition 5. Access pattern ($\varphi_5$): Given a query, which data identifiers satisfy the query criteria and which data identifiers do not satisfy the query criteria. For Access pattern, the inputs of $L$ are encrypted index $\Gamma^*$ and token $TK$, and the output of $L$ is a $n$-dimensional vector $\varphi_5 = (\beta_1, \beta_2, \cdots, \beta_n) \leftarrow L(\Gamma^*, TK)$, where $\beta_i \in \{0, 1\}$ for $i = 1, 2, \cdots, n$. If $\beta_i = 0$, it means spatial data $C_{di}$ is not in the query range, otherwise it means spatial data $C_{di}$ is in the query range.

B. Security Definition

We defined our security definition using a game-based approach that is widely used in SE scenarios. The security of our scheme can be summarized as data privacy and query privacy, either of which can be rigorously verified with Selective Chosen-Plaintext Attack (IND-CPA).

Data Privacy. Our data privacy shows that by submitting two spatial datasets $DB_0$ and $DB_1$, a computationally limited adversary $A$ is able to select a large number of ciphertext requests and token requests confined by the leakage function $L$. However, it is not computationally feasible for adversary $A$ to distinguish the two datasets.

Definition 6. IND-CPA Data Privacy. Let $\Pi =$ (Setup, IndexGen, DBEnc, QueryGen, TokenGen, Match, Dec) be a range query scheme (ARQ) based on the security parameter $\lambda$. We define the following security games between Challenger $C$ and adversary $A$:

Init. The adversary $A$ submits two datasets $DB_0$ and $DB_1$ to challenger $C$, where $DB_0 = \{D_{0,1}, D_{0,2}, \cdots, D_{0,n}\}$, $DB_1 = \{D_{1,1}, D_{1,2}, \cdots, D_{1,n}\}$. For $i = 1, \cdots, n$, $D_{0,i} \in \Delta^\tau_p$, $D_{1,i} \in \Delta^\tau_p$ and $DB_0$ and $DB_1$ satisfy $L(\Gamma^*_0) = L(\Gamma^*_1)$.

Setup. The challenger $C$ runs Setup algorithm to generate key $sk = \{k_1, k\}$.

Phase 1. The adversary $A$ adaptively submits requests of one of the following types:

- **Ciphertext Request**: For the $j$th ciphertext request, adversary $A$ submits a dataset $DB_j$, where $DB_j = \{D_{j,1}, D_{j,2}, \cdots, D_{j,n}\}$, then challenger $C$ responds with an encrypted index $\Gamma^*_j$, where $\Gamma^*_j \leftarrow DBEnc(sk, \Gamma_j)$, $\Gamma_j \leftarrow IndexGen(DB_j)$

- **Token Request**: For the $j$th token request, adversary $A$ submits a query $q_j$, where $L(\Gamma^*_0, TK_j) =$
The Challenger randomly selects a bit $b$ from \{0, 1\}, then invokes $\text{IndexGen}$ and $\text{DBEnc}$ algorithms to get index $\Gamma_b$ and encrypted index $\Gamma_b^*$, and returns $\Gamma_b^*$ to adversary.

**Phase 2.** Adversary $A$ continues to adaptively send queries to the challenger, which have the same constraints as described in Phase 1.

**Guess.** Adversary $A$ outputs a bit $b'$ as a guess of $b$.

We believe that the scheme II is secure under Selective Chosen-Plaintext Attack if for any probabilistic polynomial-time adversary $A$, the advantage $\text{Adv}_{\Pi, II}^{\text{Data}}(1^\lambda)$ of adversary $A$ is a negligible function based on $\lambda$. That is:

$$\text{Adv}_{\Pi, II}^{\text{Data}}(1^\lambda) = |Pr[b' = b] - \frac{1}{2}| \leq \text{negl}(\lambda),$$

where $\text{negl}$ is a negligible function.

**Query Privacy.** The definition of query privacy is similar to the definition of data privacy except that it submits two range queries $q_0$ and $q_1$ instead of two databases. The details are as follows:

**Definition 7. IND-SCPA Query Privacy.** Let $\Pi = (\text{Setup}, \text{IndexGen}, \text{DBEnc}, \text{QueryGen}, \text{TokenGen}, \text{Match}, \text{Dec})$ be a symmetric key range query scheme (ARQ) based on the security parameter $\lambda$. We define the following security games between Challenger $C$ and adversary $A$:

**Init.** Adversary $A$ submits two range queries $q_0$ and $q_1$ to challenger $C$, where $\mathcal{L}(TK_0) = \mathcal{L}(TK_1)$.

**Setup.** Challenger runs the Setup algorithm to generate $sk = \{k_1, k\}$.

**Phase 1.** Adversary $A$ adaptively submits one of the following types of requests:

- **Ciphertext Request:** On the $j$th ciphertext request, the adversary $A$ submits a database $DB_j$, where $DB_j = \{D_{j,1}, D_{j,2}, \ldots, D_{j,n}\}$, then Challenger $C$ responds to an encrypted index $\Gamma_j^*$, where $\Gamma_j^* \leftarrow \text{DBEnc}(sk, \Gamma_j)$, $\Gamma_j \leftarrow \text{IndexGen}(DB_j)$, and the encrypted index satisfies $\mathcal{L}(\Gamma_j, TK_0) = \mathcal{L}(\Gamma_j^*, TK_1)$.

- **Token Request:** On the $j$th token request, adversary $A$ submits query $q_j^*$, where $\mathcal{L}(TK_j, TK_0) = \mathcal{L}(TK_j^*, TK_1)$, then Challenger $C$ responds with a query token $TK_j^*$, where $TK_j^* \leftarrow \text{TokenGen}(sk, Q_j)$, $Q_j^* \leftarrow \text{QueryGen}(q_j^*)$.

**Challenge.** Challenger $C$ randomly selects a bit $b$ from \{0, 1\}, then calls the $\text{QueryGen}$ and $\text{TokenGen}$ algorithms to get the query $Q_b$ and the encrypted token $TK_b^*$, and returns the $TK_b^*$ to adversary $A$.

**Phase 2.** Adversary $A$ continues to adaptively send queries with the same constraints as described in Phase 1 to the challenger.

**Guess.** Adversary $A$ outputs a bit $b'$ as a guess of $b$.

We believe that the scheme II is secure under Selective Chosen-Plaintext Attack if for any probabilistic polynomial-time adversary $A$, the advantage of adversary $A$ is a negligible function based on $\lambda$. That is:

$$\text{Adv}_{\Pi, II}^{\text{Query}}(1^\lambda) = |Pr[b' = b] - \frac{1}{2}| \leq \text{negl}(\lambda),$$

where $\text{negl}$ is a negligible function.

**C. Security Proof**

In this paper, we adopt Hilbert curve to divide the region into atomic regions, and adopt SSW to judge whether the spatial data points are in the query range. Since the construction of Hilbert curve is unidirectional and SSW has the indistinguishability of ciphertext under selective chosen-plaintext attack, our ARQ scheme can realize data privacy and query privacy by using these mature building tools. In addition to users, LBSP and CSP are also involved in our ARQ scheme. In the following sections, we will demonstrate the security of the scheme from the perspective of building tools and participants.

**Theorem 1.** The space transformation of Hilbert curve is unidirectional. Without knowing the space transformation parameters, it is difficult to determine its position in two-dimensional space according to one-dimensional Hilbert value.

Given the order $N$, the starting point $(x_0, y_0)$, the direction $\mathcal{D}$ and the scale factor $\Theta$ of the curve, a Hilbert curve can be uniquely determined. For malicious adversaries, it is infeasible to calculate the space transformation parameters by comparing Hilbert values for all starting points.

**Brute-force attack:** Suppose the starting point of the curve is $(x_0, y_0)$, whose horizontal and vertical coordinates are represented by $l$ bits respectively. In order to get the starting point $(x_0, y_0)$, the malicious adversary needs to generate $2^l$ values on the $x$-axis and $y$-axis separately, so it is necessary to find $(x_0, y_0)$ in the $(2^l + 2^l)$ elements. In the same way, assuming the direction $\mathcal{D}$ of the curve is represented by $l$ bits, then the continuous $360^\circ$ space can be discretized into $2^l$ values. The adversary must try $2^l$ times to determine the direction of the curve. Given the scale factor, if the curve’s order is $N$, there are $(N \cdot 2^l \cdot 2^l \cdot 2^l)$ choices in the whole space. Therefore, the complexity of obtaining spatial transformation parameters by violent attack is $O(2^{3L})$, which makes the Hilbert transformation irreversible given a sufficiently large value of $l$.

**Theorem 2.** The ARQ scheme proposed in this paper can achieve data privacy if SSW has the indistinguishability of ciphertext under selective chosen-plaintext attack.

**Proof** The proof is based on a probability polynomial time (PPT) simulator which works as a challenger, it is proved that compromising the proposed scheme is equivalent to compromising the security of the building tool. The details are as follows:

**Init.** Adversary $A$ selects two databases $DB_0$ and $DB_1$ and submits them to the challenger. Both $DB_0$ and $DB_1$ contain $n$ elements.

**Setup.** The Challenger randomly selects parameters from $\mathbb{Z}_p$, $\mathbb{Z}_q$, $\mathbb{Z}_r$ and key from $1^\lambda$.

**Phase 1.** The adversary adaptively generates one of the following two requests:

- **Ciphertext request:** The adversary outputs a database $DB_j$ ($j \in \{0, 1\}$) and a target element $DB_{j*,i}$. The Challenger calls $\text{IndexGen}$ algorithm and $\text{DBEnc}$ algorithm to generate index $\Gamma_j$ and encrypted index $\Gamma_j^*$ respectively, and finally returns the encrypted target element $\Gamma_{j*,i}$ as a response.

**Phase 2.** Adversary $A$ continues to adaptively send queries with the same constraints as described in Phase 1 to the challenger.

**Guess.** Adversary $A$ outputs a bit $b'$ as a guess of $b$.

We believe that the scheme II is secure under Selective Chosen-Plaintext Attack if for any probabilistic polynomial-time adversary $A$, the advantage of adversary $A$ is a negligible function based on $\lambda$. That is:

$$\text{Adv}_{\Pi, II}^{\text{Query}}(1^\lambda) = |Pr[b' = b] - \frac{1}{2}| \leq \text{negl}(\lambda),$$

where $\text{negl}$ is a negligible function.
• **Trapdoor request:** By calling QueryGen algorithm, the adversary outputs the query $Q_0^*$ and sends it to the challenger. If $L(\Gamma_0^*, TK_1^*) = L(\Gamma_1^*, TK_2^*)$, the challenger calls TokenGen algorithm to generate the token $TK_2^*$ and returns it to the adversary as a response.

**Challenge.** Challenger randomly selects a bit $b$ from $\{0, 1\}$, calls IndexGen algorithm to generate the index $\Gamma_b^*$, and then calls the DBEnc algorithm to get the encrypted index $\Gamma_b^*$.

**Phase 2.** The adversary continues to adaptively send two requests as described in phase 1 to the challenger.

**Guess.** The adversary outputs a bit $b'$ as the guess of $b$.

The ARQ scheme is successfully simulated by a PPT simulator. It shows that if a PPT adversary can break the scheme, it must be able to break the SSW encryption algorithm under selective chosen-plaintext attack.

**Theorem 3.** The ARQ scheme proposed in this paper can achieve query privacy if SSW is IND-SCPA query secure.

The proof of query privacy is similar to that of data privacy. Due to space constraints, we skipped the full proof.

**Theorem 4.** The ARQ scheme proposed in this paper can realize privacy preservation against untrusted LBSP.

**Proof** In the process of a range query based on a location, LBSP receives the user’s request for the atomic region coding of a region $R$. According to this request, LBSP can only obtain the information about user’s region, but can’t judge the specific location and the size of the range query. Therefore, ARQ scheme can protect location privacy and query privacy of users from LBSP.

**Theorem 5.** The ARQ scheme proposed in this paper can realize privacy preservation against untrusted CSP.

**Proof** After LBSP outsources the spatial dataset to CSP, the user performs the range query mainly by interacting with CSP. On the one hand, CSP gets the encrypted index from the LBSP. As defined by leakage function $L$ in 6.1, CSP can obtain Size pattern and Structure pattern based on the encrypted index, that is, CSP can obtain the size of the dataset and the number and size of atomic regions containing spatial data. Because of the monotony of the Hilbert curve, it is unfeasible for CSP to determine the specific location of the spatial data by analyzing atomic region codes. Therefore, ARQ can protect the data privacy of LBSP against untrusted CSP. On the other hand, CSP receives encrypted query requests from user and thus obtains a Query-size pattern. As mentioned above, due to the monotony of Hilbert curve and confidentiality of SSW encryption algorithm, CSP cannot know the specific location of range query, so ARQ can protect user’s query privacy. To sum up, The ARQ scheme can protect the data privacy of LBSP and user’s query privacy against untrusted CSP.

**VII. Performance Analysis**

**A. Complexity Analysis**

We will evaluate the effectiveness of the scheme from the perspective of the participants, based on the computation cost and communication cost of the user, LBSP and CSP in the whole operation process of the scheme.

1) **Computation cost:** In the scheme, the user performs the query generation, token generation and decryption phases. After interaction with LBSP, the user obtains atomic region codes of the query area $R$, determines atomic region sets to be queried and query vector according to the query location and radius, which are hashed and encrypted respectively by user and sent to CSP. The processing time of this process is mainly spent on hashing the atomic region sets and encrypting the query vector. Compared with encryption operation, the calculation time of hashing operation is negligible, while the complexity of encrypting query vectors is related to the size of query range. The larger the query range is, the more atomic regions intersect the query range, whose computational complexity is $O(r^2\alpha)$, where $\alpha$ is the average number of query vectors in an atomic region. Therefore, the computational complexity of query generation and token generation is $O(r^2\alpha)$. In the decryption stage, the user needs to decrypt the encrypted dataset in the query range. The processing time of this stage is related to the amount of data in the query range. The size of the query range and the total amount of spatial data in the region $R$ will affect the amount of data in the query range. The larger the query range is, the more data there is in the atomic region. Similarly, when the size of the query range is constant, the more data the region $R$ includes, the more spatial data will distribute in the query range. In summary, the user’s computational complexity in decryption stage is $O(nr^2)$. Therefore, the calculation cost of the user is $O((n + \alpha)r^2)$, where $r$ is the query radius, and $n$ is the data amount in region $R$.

In the scheme, LBSP performs initialization, index generation and database encryption phases. Among them, the processing time of initialization phase is mainly spent on the partition of region $R$, which only needs to be executed once and has a little impact on the whole computation cost of LBSP. The processing time of index generation and database encryption phase is related to the amount of spatial data, and its computational complexity is $O(n)$. Therefore, the computation cost of LBSP is $O(n)$.

In the scheme, CSP performs the matching and re-encryption phase. The processing time of matching phase is mainly spent on comparing the spatial data with query vector, and the cost of which is related to the data amount in the atomic region interacting with the query range. The higher the order of Hilbert curve is, the more the number of atomic regions is, and the less the amount of spatial data is in each atomic region, that is, the amount of spatial data in the atomic region is inversely proportional to the number of atomic regions $2^{2N}$, but with the increase of the total amount $n$ of spatial data in region $R$, the amount of spatial data in the atomic region will also increase. In the re-encryption stage, CSP needs to re-encrypt the spatial data in the query range. The computational complexity of this stage is related to the query size $r^2$ and the total amount of spatial data in region $R$. On the one hand, the larger the query range is, the more spatial data it contains. On the other hand, the more the total amount of spatial data is, the more spatial data is in the query range. Therefore, the computation cost of CSP is $O(n \cdot 4^{-N} + nr^2)$. 

**Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on March 26,2021 at 07:36:36 UTC from IEEE Xplore. Restrictions apply.**
2) Communication cost: The communication cost between user and CSP is firstly analyzed. In the whole process of range query, the user outputs the token $TK$ with a fixed size $|TK|$ to CSP, and the user receives the spatial data in the query range from CSP. If the size of each spatial data is set as $|D|$, then the size of spatial data in the query range is $(n'|D|)$, where $n'$ represents the total number of spatial data in query range. Therefore, the user’s communication cost is $|(TK| + n'|D|)$, LBSP sends encrypted data set to CSP, and thus its communication cost is $(n|D|)$. In summary, the communication cost of CSP is $|(TK| + (n'+ n)|D|)$.

We introduces the meaning of variables involved in the performance analysis in Table II, and summarize the computation time and communication time of all entities in Table III.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>Radius of the range query</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>The average number of query vectors in an atomic region</td>
</tr>
<tr>
<td>$N$</td>
<td>Order of Hilbert curve</td>
</tr>
<tr>
<td>$n$</td>
<td>Total number of spatial data in a region</td>
</tr>
<tr>
<td>$n'$</td>
<td>Total number of spatial data in a circle range</td>
</tr>
<tr>
<td>$</td>
<td>TK</td>
</tr>
<tr>
<td>$</td>
<td>D</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Length of location vector and query vector</td>
</tr>
<tr>
<td>$</td>
<td>\epsilon_G</td>
</tr>
<tr>
<td>$t_p$</td>
<td>Time cost for a bilinear mapping</td>
</tr>
<tr>
<td>$t_e$</td>
<td>Time cost for an exponential operation in $G_p$</td>
</tr>
<tr>
<td>$t'_e$</td>
<td>Time cost for an exponential operation in $G_q$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III EFFICIENCY ANALYSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>user</td>
</tr>
<tr>
<td>LBSP</td>
</tr>
<tr>
<td>CSP</td>
</tr>
</tbody>
</table>

B. Performance Evaluation

In this paper, Hilbert curve and SSW encryption algorithm are used to construct a range query scheme. In this part, we will first analyze the efficiency of the scheme from the perspective of building tools. Then we evaluate the performance of our scheme over encrypted data in several aspects, including encryption time, token generation time, token size, and search time. The two main parameters that impact performance are Hilbert curve’s order $N$ and query size $r$. The experiment was performed on a local 63-bit PC with Intel Core i5-3230m processor at 2.6GHz and 8GB RAM. MATLAB R2014a was used to construct Hilbert Curve, and the Pairing Based Cryptography (PBC) library and GNU Multiple Precision (GMP) library are used to implement the pairing group operations, so as to test the running time of cryptography operation in the scheme. We used a real weibo check-in dataset and conducted experiments on the dataset in Beijing city, which contains 59,780 data points. This dataset size is in line with the mainstream dataset size (for example, see [7] and [44]). The distribution of the test dataset is shown in Fig. 9.

Fig. 9. Distribution of locations in the Beijing city

1) Hilbert Curve: Hilbert curve is used to divide the region into atomic regions in this paper, which realize fine-grained query and sub-linear search time. Fig. 10 describes the calculation time needed to construct the Hilbert curve of different orders. From the picture we can see that as the order of Hilbert curve becomes larger, it takes more time to construct the curve, but this time is relatively small compared with the whole scheme. When the order is 10, it only takes 0.05ms to construct the Hilbert curve. Therefore, it is efficient to use Hilbert curve for region division.

Fig. 10. Time cost for constructing Hilbert curve

2) SSW Encryption algorithm: SSW encryption algorithm is used to encrypt spatial data, generate token and search on the encrypted spatial data to determine the relationship between spatial data and query range, which is the main construction tool of our ARQ scheme. According to the construction of SSW encryption algorithm in the literature [8], we analyze the efficiency of SSW algorithm in detail. In Table I, $|\epsilon_G|$ represents the size of the elements in group $G$. $t_p$, $t_e$ and $t'_e$ respectively represent the time required for a bilinear mapping, an exponential operation in group $G_p$, and an exponential operation in group $G_q$. Table IV describes the size of key, ciphertext and token and the computation time required for different stages in SSW, where $\omega$ represents the length of location vector and query vector.

The impact of Hilbert Curve’s Order When the region is divided, the order of Hilbert curve will affect the query efficiency. The higher the order is, the more the number of atomic regions is, and the less spatial data each atomic region contains. Therefore, for a specific query range, fewer query comparison operations need to be performed. For a spatial dataset of about 60000, Fig. 11 depicts the relationship
between the order of Hilbert curve and the data amount in the atomic region. It can be seen from the figure that when $N = 2$, the average number of spatial data in the atomic region is 3736; when $N = 7$, the average data amount in the atomic region is 4. Therefore, by using Hilbert curve to divide the region into atomic regions, the query efficiency can be improved to sublinear.

![Fig. 11. The impact of Hilbert curve’s order](image)

The order $N$ will influences the edge length $d$ of atomic region: a larger $N$ corresponds to a smaller $d$. Since the length $\omega$ of location vector and query vector is related to $d$, a larger $N$ corresponds to a shorter $\omega$. Table V describes the impact of $N$ on the encryption time. Note that in the process of encrypting spatial datasets, the time for hashing atomic region coding can be ignored compared with encrypting location vectors. When $N = 7$, given the area of Beijing, the edge length $d$ is about 1000.86$m$. At this time, the length $\omega$ of location vector is 33, and it takes 823.171$s$ to encrypt our dataset. For the same dataset, if we choose a larger $N$, the less time it takes to encrypt the location vector. For example, when $N = 9$, $d$ is about 250.215$m$, and it takes 428.025$s$ to encrypt the datasets. Although encrypting the spatial dataset takes some time, it is only a one-time cost.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$d$ (m)</th>
<th>$\omega$</th>
<th>Encryption Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64055m</td>
<td>33</td>
<td>823.171s</td>
</tr>
<tr>
<td>7</td>
<td>1000.86m</td>
<td>21</td>
<td>527.260s</td>
</tr>
<tr>
<td>8</td>
<td>500.43m</td>
<td>19</td>
<td>477.642s</td>
</tr>
<tr>
<td>9</td>
<td>250.22m</td>
<td>17</td>
<td>428.025s</td>
</tr>
</tbody>
</table>

Given the query size, the order $N$ can affect the token generation time. Table VI describes the impact of $N$ on token generation time under $r = 1000m$. When $N = 7$, it takes about 11.592$s$ to generate a query vector and 1.609$s$ to generate a token corresponding to a given query range. When $N = 9$, it takes about 9.384$s$ to generate a query vector and 11.148$s$ to generate a token corresponding to a given query range. It can be seen from the table that the computation time of query vector is negatively correlated with $N$. However, with the increase of $N$, the number of atomic regions that intersect with the query range increases, which leads to a significant rise of the whole token generation time.

<table>
<thead>
<tr>
<th>Query range</th>
<th>$N$</th>
<th>Time for a query vector</th>
<th>Time for token</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>11</td>
<td>11.592$s$</td>
<td>1.609$s$</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>10.488$s$</td>
<td>4.531$s$</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9.384$s$</td>
<td>11.148$s$</td>
</tr>
</tbody>
</table>

It is worth noting that when we use the cryptographic primitives PRF and SSW to generate tokens for the query range, we need to enumerate all possible data points inside the query range in the plaintext domain, and then generate the corresponding query vectors in plaintext. Since this sub-step is completely calculated in plaintext, this process will hardly affect the performance of token generation.

Besides token generation time, the order also affects the query time. In our scheme, only encrypted data points in the same atomic region as the token are retrieved. As shown in Table VII, when $N = 7$, there are about 10 data points in each atomic region for a given dataset, and the query time is 6.970$s$. When $N = 9$, there are about 3 data points in each atomic region, and it takes 1.711$s$ to search.

<table>
<thead>
<tr>
<th>Query range</th>
<th>$N$</th>
<th>$\omega$</th>
<th>Data amount in one atomic region</th>
<th>Search time</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>21</td>
<td>10</td>
<td>6.970$s$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>6</td>
<td>3.802$s$</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>3</td>
<td>1.711$s$</td>
<td></td>
</tr>
</tbody>
</table>

The impact of $N$ on token size is shown in Table VIII, for a given query range of $r = 1000m$, when $N = 7$, the size of a query vector is 2.178$KB$, and the token size corresponding to the query range is 313.632$KB$. However, when $N = 9$, the size of a query vector is 1.782$KB$, and the token size corresponding to the same query range is 2.067$MB$.

<table>
<thead>
<tr>
<th>Query range</th>
<th>$N$</th>
<th>$\omega$</th>
<th>Size of a query vector</th>
<th>Token size</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>21</td>
<td>2.178$KB$</td>
<td>313.632$KB$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>1.98$KB$</td>
<td>855.36$KB$</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>1.782$KB$</td>
<td>2.067$MB$</td>
<td></td>
</tr>
</tbody>
</table>

The impact of query size $r$. In addition to the order $N$, the query size $r$ also affects the token generation time and query time. Given $N = 8$ and $\omega = 19$, we choose four different query ranges: 500$m$, 1000$m$ and 2000$m$. The token generation time and query time are shown in Table IX.

<table>
<thead>
<tr>
<th>Query range</th>
<th>$N$</th>
<th>$\omega$</th>
<th>Size of a query vector</th>
<th>Token size</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>21</td>
<td>2.178$KB$</td>
<td>313.632$KB$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>1.98$KB$</td>
<td>855.36$KB$</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>1.782$KB$</td>
<td>2.067$MB$</td>
<td></td>
</tr>
</tbody>
</table>

C. Comparison with other solutions

We compare our scheme with the latest privacy-preserving range query scheme (CRSE[15] and FastGeo[7], respectively),
where CRSE can query the spatial data in the circular range over encrypted dataset and FastGeo can query the spatial data in the geometric range over encrypted dataset. We comprehensively compare the time of encryption, token generation and search, and compare the complexity and accuracy of the above schemes as shown in Table IX below. The query size tested below is 100m and the dataset size is 1000. The length of the vector in CRSE is 4, and that in FastGeo is 100. In our scheme, we use 8-order Hilbert curve to partition the region and hence the length of vector is 19.

From the table we can see that the query complexity of CRSE is linear, and the search time increases linearly with the increase of data amount. Compared with CRSE, our scheme manages the encrypted dataset and query range by atomic region, so that the search process can be processed in a distributed way. Therefore, the search efficiency of ours is significantly higher than that of CRSE. In addition, scheme CRSE can only query the circular range, and our scheme can be extended to query any shapes besides circular range.

In FastGeo, the length of vector depends on the size of dataset, while the length of vector in our scheme is related to the side length of atomic region, so the encryption time and token generation time of FastGeo are significantly higher than ours. Since FastGeo uses a two-layer structure to build indexes, its query time is sublinear and does not increase linearly as the dataset increases. Region division and the construction of vector designed in this paper enable that our scheme can achieve accurate and efficient nearest neighbor or range query such as “querying the nearest hospital around me” is also a common location-based query. Under the condition of protecting data privacy and query privacy, how to implement a secure and efficient range query without a third-party server usually cannot resist collusion attacks, which poses new threats to LBSP’s data privacy and user’s query privacy. Our next work will consider how to implement a secure and efficient range query without a third-party server.

In addition to range query, nearest neighbor or k-nearest neighbor query such as “querying the nearest hospital around me” is also a common location-based query. Under the condition of protecting data privacy and query privacy, how to design accurate and efficient nearest neighbor or k-nearest neighbor query scheme will also be the research direction of our next work.

### REFERENCES


### TABLE IX

<table>
<thead>
<tr>
<th>$N = 8$, $\omega = 19$</th>
<th>Query range</th>
<th>Token Generation Time</th>
<th>Search Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r = 500m$</td>
<td>1.510s</td>
<td>3.802s</td>
<td></td>
</tr>
<tr>
<td>$r = 1000m$</td>
<td>4.551s</td>
<td>5.619s</td>
<td></td>
</tr>
<tr>
<td>$r = 1500m$</td>
<td>7.551s</td>
<td>11.651s</td>
<td></td>
</tr>
<tr>
<td>$r = 2000m$</td>
<td>12.082s</td>
<td>15.206s</td>
<td></td>
</tr>
</tbody>
</table>

**Query range token generation and search**

**VIII. CONCLUSION**

In this paper, we propose a privacy-preserving range query scheme for outsourced LBS in IoT. By using Hilbert curve to divide the region into atomic regions and using SSW encryption algorithm to judge the relationship between spatial data and query range, our ARQ scheme can achieve accurate range query on the premise of preserving data privacy and query privacy, and further achieve sublinear search time and effective update of spatial data. Privacy preservation schemes that involve third-party servers usually cannot resist collusion attacks, which poses new threats to LBSP’s data privacy and user’s query privacy. Our next work will consider how to implement a secure and efficient range query without a third-party server.
## PERFORMANCE COMPARISON

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Encryption</th>
<th>Token</th>
<th>Search</th>
<th>Complexity</th>
<th>Accuracy</th>
<th>Range type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRSE</td>
<td>5.61s</td>
<td>329.47ms</td>
<td>98.65s</td>
<td>linear</td>
<td>10m</td>
<td>circle</td>
</tr>
<tr>
<td>FastGeo</td>
<td>19.67s</td>
<td>550.38ms</td>
<td>13.67s</td>
<td>sublinear</td>
<td>10m</td>
<td>any shape</td>
</tr>
<tr>
<td>Ours</td>
<td>3.79s</td>
<td>377.57ms</td>
<td>3.80s</td>
<td>sublinear</td>
<td>1m</td>
<td>any shape</td>
</tr>
</tbody>
</table>


Zhaoman Liu received the B.S. degree in information management and information system from the Shandong Normal University, Jinan, China, in 2017. She is currently a postgraduate of Information Science and Engineering of Shandong Normal University. Her research interest includes privacy preservation in location-based service and cloud computing security.
Lei Wu received the Ph.D. degree in applied mathematics from Shandong University, China in 2009. He is currently an associate professor at Shandong Normal University, China. His research interest includes cryptology and cloud computing security.

Weizhi Meng obtained his Ph.D. degree in Computer Science from the City University of Hong Kong (CityU), Hong Kong. Prior to joining DTU, he worked as a research scientist in Institute for Infocomm Research, A*Star, Singapore. He won the Outstanding Academic Performance Award during his doctoral study, and is a recipient of the Hong Kong Institution of Engineers (HKIE) Outstanding Paper Award for Young Engineers/Researchers in both 2014 and 2017. His primary research interests are cyber security and intelligent technology in security, including intrusion detection, smartphone security, biometric authentication, HCI security, IoT security and blockchain. He is a senior member of IEEE.

Hao Wang received his Ph.D. degree in computer science from Shandong University, China, in 2012. He is currently an associate professor in Shandong Normal University. His primary interest is public key cryptography, in particular, designing cryptographic primitives and provable security. At present, he is focusing on attribute based cryptography, security in cloud computing, and blockchain.

Wei Wang received the B.S. degree in software engineering from the Shandong Jianzhu University, Jinan, China, in 2018. He is currently a graduate student of Information Science and Engineering of Shandong Normal University. His research interest includes privacy preservation in the Internet of Vehicles and fog computing.