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
Peer reviewed version

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

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 issuitable 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

This work was supported in part by the Natural Science Foundation of Shandong Province under Grant ZR2020MF056 and Grant ZR2020KF011, in part by the National Natural Science Foundation of China under Grant 62071280, and in part by the Major Scientific and Technological Innovation Project of Shandong Province under Grant 2020CXGC010115. *(Corresponding author: Lei Wu.)*

Z Liu, H Wang and W Wang are with the School of Information Science and Engineering, Shandong Normal University, Jinan 250307, China, and also with Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology, Jinan 250307, China (E-mail: zhaomanliu1995@163.com; wanghao@sdnu.edu.cn; 995251486@qq.com).

L Wu is with the School of Information Science and Engineering, Shandong Normal University, Jinan 250307, China, with Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology, Jinan 250307, China, and also with Shandong Provincial Key Laboratory for Software Engineering, Jinan 250307, China (E-mail: wulei@sdnu.edu.cn).

W Meng is with the Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Denmark.

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

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*) [1] can only support equality testing; Order-Preserving Encryption (*OPE*) [2] only supports comparison; Partial Homomorphic Encryption (*PHE*) can only calculate addition (such as Pailliar encryption [3]) or multiplication (such as Elgamal encryption [4]). In principle, Full Homomorphic Encryption (*FHE*) [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 LBSP's data privacy and the user's query privacy to the cloud server. Compared with our previous scheme [6], this scheme can realize accurate range query. Compared with scheme [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 [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-SCPA). 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 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 [9, 10] essentially provide a solution that supports the search of rectangular range. Specifically, Boneh et al. [9] and Shi et al. [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 [12, 13] or *kd*-tree [14], the schemes [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. [15] proposed a scheme that could retrieve data points within the circular range from the encrypted data. Zhu et al. [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. [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 [18] and predicate encryption [8]. The scheme in [7] improves the search complexity by designing a new form of equal-vector [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. [19] proposed a generalized geometric range query scheme for encrypted datasets based on *ASPE* [20] and geometric

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

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.

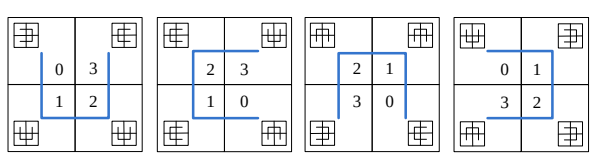


Fig. 1. Four transformations of Hilbert Curve

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

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

TABLE I
COMPARISON OF SCHEMES ON RANGE QUERY WITH PRIVACY PRESERVATION

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

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

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

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(1^λ): Given the security parameter λ, 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 α 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

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

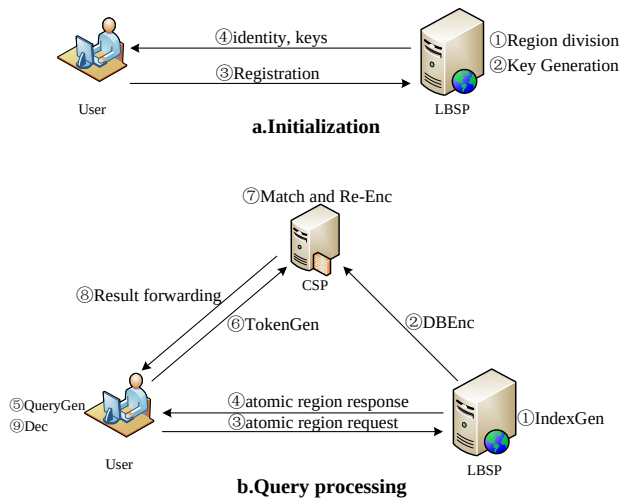
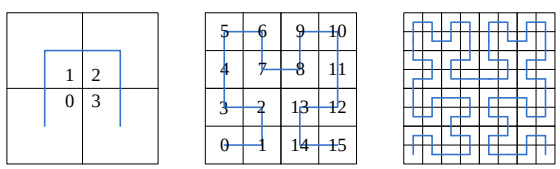


Fig. 2. System model for outsourced range query

curve, 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.



(a) First-order (b) Second-order (c) Third-order

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 $\vec{v}$ 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 atomiic 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.

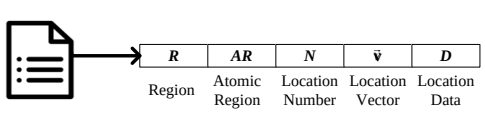


Fig. 4. The storage structure of spatial data

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 $\{\vec{u}\}$ 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 $\{\vec{u}\}$ 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}.

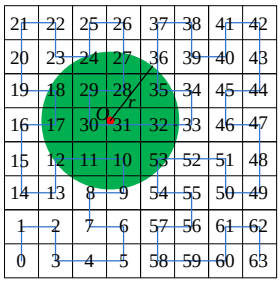


Fig. 5. The storage structure of spatial data

D. Threat Model

Our threat model is basically consistent with other work in this area. CSP and LBSP are perceived as honest but curious, that is, they will abide by the agreement honestly, but they also want to analyze and infer the private information of other entities from the obtained data. Specifically, according to the user's request, LBSP will respond honestly to atomic region coding but also tries to infer the user's query range. In addition, CSP will honestly match the query initiated by the user. At the same time, it also wants to infer the user's query range and information of spatial data according to the content requested by the user as well as the encrypted data sent by LBSP. This paper assumes that there is no collusion between LBSP and CSP in the query process. Based on what the cloud server has learned, we summarize the two threat models as follows.

1) Known ciphertext model: This threat model refers to ciphertext-only attack. CSP obtains encrypted atomic region encoding and encrypted query vectors from the user, and obtains encrypted location vectors and encrypted dataset from the LBSP. According to the above encrypted information, CSP may attempt to deduce about the user's location and the concrete location of spatial data.

2) Known background model: CSP can record and analyze queries since user may continuously send requests for range queries. If CSP learns any valuable information from the recorded query, it may deduce the user's approximate location and trajectory. In addition, it is possible for CSP to analyze the construction of Hilbert curve through the atomic region sets submitted by the user, and then infer the user's location. CSP stores the location vector sent by LBSP and performs query

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

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 $\mathcal{G}(1^\lambda)$ to obtain $(p, q, r, s, \mathbb{G}, \mathbb{G}_T, e)$, where p, q, r, s are random prime number and $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r \times \mathbb{G}_s$ is a *N*-order composite group ($N = pqrs$), and then LBSP selects generators g_p, g_q, g_r, g_s from group $\mathbb{G}_p$, $\mathbb{G}_q$, $\mathbb{G}_r$, $\mathbb{G}_s$ respectively. For $i = 1$ to ω, LBSP selects $h_{1,i}$, $h_{2,i}$, $u_{1,i}$, $u_{2,i} \in \mathbb{G}_p$ uniformly and randomly, where ω 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 $rk_{p\to i}$, which is used to re-encrypt the spatial data. Finally, LBSP sends *param* = $(id_i, pk_i, sk_i, rk_{p\to i}, sk)$ 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

Algorithm 1 *IndexGen*(*DB*)$\rightarrow \Gamma$

$\Gamma \leftarrow$null
$\{AR_1, AR_2, \cdots, AR_m\} \leftarrow$*Hilbert*(*R*)
for $i \leftarrow 1, 2, \cdots, m$ **do**
  $\Gamma_i \leftarrow Linklist.Init()$
  $\Gamma_i \leftarrow Linklist.Append(\Gamma_i, R, AR_i)$
  for $j \leftarrow 1, 2, \cdots, |AR_i|$ **do**
    $\vec{v}_j \leftarrow \{0, 1\}^\omega$
    $\Gamma_{ij} \leftarrow (j, \vec{v}_j, D_j)$
    $\Gamma_i \leftarrow Linklist.Append(\Gamma_i, \Gamma_{ij})$
  end for
  $\Gamma \leftarrow \Gamma \cup \{\Gamma_i\}$
end for
return Γ

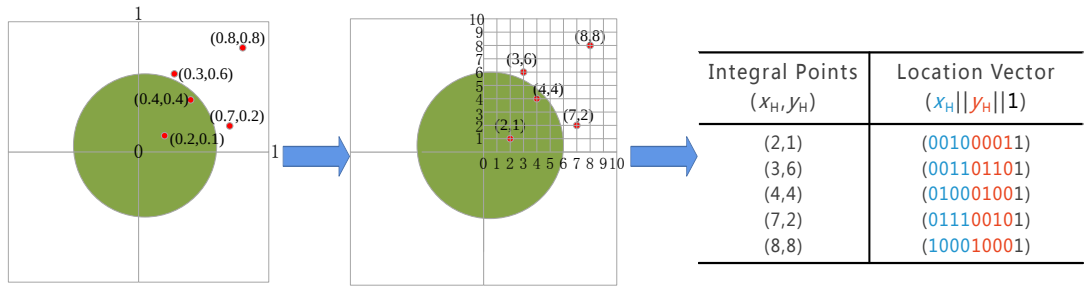
to represent the location vector, which obviously shortens the vector length and ensures the unification of the vector length in different atomic regions, thus avoiding the inference attack caused by the different vector length. Suppose the edge length of atomic region is an integer d, and the data points 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 $\lceil l \log_2 10 + \log_2 d \rceil$ bits. Among them, The side length d of the atomic region is inversely proportional to the order N and directly proportional to the length S of the region, which can be expressed as $d = S/2^N$. We use $\omega = 2\lceil l \log_2 10 + \log_2 d \rceil + 1$ 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 Γ 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

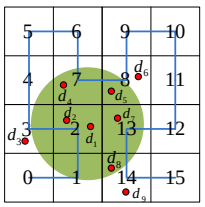
This phase is performed by LBSP. Since the location of terminal device is confidential information of LBSP, in order to protect data's privacy, LBSP needs to further encrypt the index Γ. Specifically, LBSP will encrypt the atomic region coding, location vector and data content in the index. Firstly,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

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



Integral Points (x_H, y_H)	Location Vector $(x_H \| y_H \| 1)$
(2,1)	(001000011)
(3,6)	(001101101)
(4,4)	(010001001)
(7,2)	(011100101)
(8,8)	(100010001)

Fig. 6. The form of data points and location vector in one atomic region after integerization



Circular Range and dataset in region *R*

R	**2**	**1**	$\vec{v}_1$	**(x_1, y_1)**	**2**	$\vec{v}_2$	**(x_2, y_2)**
R	**3**	**1**	$\vec{v}_3$	**(x_3, y_3)**			
R	**7**	**1**	$\vec{v}_4$	**(x_4, y_4)**			
R	**8**	**1**	$\vec{v}_5$	**(x_5, y_5)**	**2**	$\vec{v}_6$	**(x_6, y_6)**
R	**13**	**1**	$\vec{v}_7$	**(x_7, y_7)**			
R	**14**	**1**	$\vec{v}_8$	**(x_8, y_8)**	**2**	$\vec{v}_9$	**(x_9, y_9)**

The index Γ of dataset *D*

Fig. 7. Index formed by spatial data within region *R*

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, α, $\beta \in \mathbb{Z}_N$, $S_1, S_2 \in \mathbb{G}_s$, $R_{1,i}$, $R_{2,i} \in \mathbb{G}_r$ 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 $C = S_1 \cdot g_p^a$, $C_0 = S_2 \cdot g_p^b$, and for $i = 1, \cdots, \omega$, $C_{1,i} = h_{1,i}^a \cdot u_{1,i}^b \cdot g_q^{\alpha v_i} \cdot R_{1,i}$, $C_{2,i} = h_{2,i}^a \cdot u_{2,i}^b \cdot g_q^{\beta v_i} \cdot R_{2,i}$. In addition, the encrypted data content C_D are calculated, where $C_D \leftarrow Enc(pk_p, D)$.

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

Algorithm 2 *DBEnc*(sk, Γ)→ Γ^*

$\Gamma^* \leftarrow$null
extract k_1, k from sk
for $i \leftarrow 1, 2, \cdots, m$ **do**
  extract AR_i from Γ_i
  $\Gamma_i^* \leftarrow Linklist.Init(\ )$
  $H_i \leftarrow H(k_1, AR_i)$
  $\Gamma_i^* \leftarrow Linklist.Append(\Gamma_i^*, R, H_i)$
  for $j \leftarrow 1, 2, \cdots, |AR_i|$ **do**
    $[\vec{v_j}] \leftarrow$*SSW.Enc*$(k, \vec{v_j})$
    $C_{D_j} \leftarrow$*Enc*(pk_p, D_j)
    $\Gamma_{ij}^* \leftarrow (j, [\vec{v_j}], C_{D_j})$
    $\Gamma_i^* \leftarrow$*Linklist.Append*$(\Gamma_i^*, \Gamma_{ij}^*)$
  end for
  $\Gamma^* \leftarrow \Gamma^* \cup \{\Gamma_i^*\}$
end for
return Γ^*

D. Query Generation

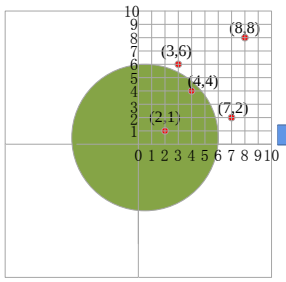
This phase is accomplished by user and LBSP. Before requesting range query, user u_i first asks LBSP for the division mode of region *R*. By default, user and LBSP share the hierarchical pattern of "country-province-city-region". After receiving the request from user u_i, LBSP encrypts the division mode 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 $CR = (CR_1, CR_2)$ 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 $\omega = 2\lceil l \log_2 10 + \log_2 d \rceil + 1$ 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 -ω_1, where ω_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_{CR1}, H_{CR2}). Secondly, calculate the encrypted query vector $[\vec{u}]$ with the key *k*. For query vector $\vec{u} \in \{\{-1, 0, 1\}^{\omega-1} || -\omega_1\}$, the user randomly selects $f_1, f_2 \in \mathbb{Z}_N$, $r_{1,i}, r_{2,i} \in \mathbb{Z}_N$ for $i = 1, \cdots, \omega$, $R_1, R_2 \in \mathbb{G}_r$,

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



Points in circular range	Merge the points (1st round)	Merge the points (2nd round)	query vector
(0,0)(0,1)(0,2)(0,3)	(0000, 00**)	(00**, 00**)	(-1-100-1-1000)
(0,4)(0,5)	(0000, 010*)	(00**, 010*)	(-1-100-11-10-1)
(0,6)	(0000, 0110)	(000*, 0110)	(-1-1-10-111-1-2)
(1,0)(1,1)(1,2)(1,3)	(0001, 00**)	(010*, 00**)	(-11-10-1-100-1)
(1,4)(1,5)	(0001, 010*)	(0100, 0100)	(-11-1-1-11-1-1-2)
(1,6)	(0001, 0110)	(0110, 000*)	(-111-1-1-1-1-2)
(2,0)(2,1)(2,2)(2,3)	(0010, 00**)		
(2,4)(2,5)	(0010, 010*)		
(3,0)(3,1)(3,2)(3,3)	(0011, 00**)		
(3,4)(3,5)	(0011, 010*)		
(4,0)(4,1)(4,2)(4,3)	(0100, 00**)		
(4,4)	(0100, 0100)		
(5,0)(5,1)(5,2)(5,3)	(0101, 00**)		
(6,0)(6,1)	(0110, 000*)		

Fig. 8. Construction of query vector in one atomic region

Algorithm 3 *QueryGen*$(q)\rightarrow Q$

$CR_1 \leftarrow$null
$CR_2 \leftarrow$null
for $i \leftarrow 1, 2, \cdots, m$ **do**
  if $AR_i \in q$ **then**
    $CR_1 \leftarrow CR_1 \cup \{AR_i\}$
  else if $AR_i \cap q \neq \emptyset$ **then**
    select all $(x, y) \in AR_i \cap q$
    transform (x, y) into $\vec{u}$
    $\vec{u} \leftarrow \{-1, 0, 1\}^{\omega-1} || -\omega_1$
    $CR_2 \leftarrow CR_2 \cup \{AR_i, \{\vec{u}\}\}$
  end if
end for
$CR \leftarrow \{CR_1, CR_2\}$
$Q \leftarrow (R, CR)$
return Q

$S_{1,i}, S_{2,i} \in \mathbb{G}_s$ for $i = 1, \cdots, \omega$, and computes the encrypted query vector

$$[\vec{u}] = (K, K_0, \{K_{1,i}, K_{2,i}\}_{i=1}^{\omega}),$$

where $K = R_1 \cdot \prod_{i=1}^{\omega} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}$, $K_0 = R_2 \cdot \prod_{i=1}^{\omega} u_{1,i}^{-r_{1,i}} \cdot u_{2,i}^{-r_{2,i}}$, and $K_{1,i} = g_p^{r_{1,i}} \cdot g_q^{f_1 u_i} \cdot S_{1,i}$, $K_{2,i} = g_p^{r_{2,i}} \cdot g_q^{f_2 u_i} \cdot S_{2,i}$ for $i = 1, \cdots, \omega$.

Finally, the user gets the token $TK = (R, H_{CR})$ and sends $(id_i, rk_{p\rightarrow i}, TK)$ to CSP to request the spatial data within the query range. A detailed description of this phase is shown in algorithm 4 as follows.

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_{CR1} with the atomic region codes in encrypted index Γ^*, and saves the encrypted spatial data from the matched atomic regions to the set *Res*. Then, CSP matches the set H_{CR2} with the atomic region codes in encrypted index Γ^*. With regard to the matched atomic regions, CSP computes $e(C, K) \cdot e(C_0, K_0) \cdot \prod_{i=1}^{\omega} e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}) \stackrel{?}{=} 1$ 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*.

Algorithm 4 *TokenGen*$(sk, Q)\rightarrow TK$

$H_{CR_1} \leftarrow$null
$H_{CR_2} \leftarrow$null
extract *CR* from Q
for all $AR \in CR_1$ **do**
  $H_{CR_1} \leftarrow H_{CR_1} \cup \{H(k_1, AR)\}$
end for
for all $AR \in CR_2$ **do**
  $\{[\vec{u}]\} \leftarrow SSW.GenToken(k, \{\vec{u}\})$
  $H_{CR_2} \leftarrow H_{CR_2} \cup \{H(k_1, AR), \{[\vec{u}]\}\}$
end for
$H_{CR} \leftarrow \{H_{CR_1}, H_{CR_2}\}$
$TK \leftarrow (R, H_{CR})$
return TK

Finally, CSP re-encrypts the dataset *Res* to get result set Res' and sends Res' to user, where $Res' \leftarrow ReEnc(rk_{p\rightarrow i}, Res)$. The algorithm description for this stage is shown in algorithm 5.

Algorithm 5 *Match*$(\Gamma^*, TK)\rightarrow Res$

Res$\leftarrow$null
extract $\Gamma_1^*, \Gamma_2^*, \cdots, \Gamma_m^*$ from Γ^*
extract H_{CR} from *TK*
for $i \leftarrow 1, 2, \cdots, m$ **do**
  extract H_i from Γ_i^*
  if $H_i \in H_{CR_1}$ **then**
    Flag=1
    for $j \leftarrow 1, 2, \cdots, |H_i|$ **do**
      Res$\leftarrow$*Res*$\cup\{C_{D_j}\}$
    end for
  end if
  if $H_i \in H_{CR_2}$ **then**
    for $j \leftarrow 1, 2, \cdots, |H_i|$ **do**
      if *SSW.Query*$(\{[\vec{u}]\}, [\vec{v_j}]) = 1$ **then**
        Flag=1
        Res$\leftarrow$*Res*$\cup\{C_{D_j}\}$
      end if
    end for
  end if
end for
return *Res*

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

G. Decryption

This phase is performed by the user. After receiving the re-encrypted result set Res', the user needs to decrypt it with his 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 $E = e(C, K) \cdot e(C_0, K_0) \cdot \prod_{i=1}^{\omega} e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i})$ 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:

$$
\begin{aligned}
E &= e(C, K) \cdot e(C_0, K_0) \cdot \prod\nolimits_{i=1}^{\omega} e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}) \\
&= e(S_1 \cdot g_p^a, R_1 \cdot \prod\nolimits_{i=1}^{\omega} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}) \cdot e(S_2 \cdot g_p^b, R_2 \cdot \\
&\quad \prod\nolimits_{i=1}^{\omega} u_{1,i}^{-r_{1,i}} \cdot u_{2,i}^{-r_{2,i}}) \cdot \prod\nolimits_{i=1}^{\omega} e(h_{1,i}^a \cdot u_{1,i}^b \cdot g_q^{\alpha v_i} \cdot R_{1,i}, \\
&\quad g_p^{r_{1,i}} \cdot g_q^{f_1 u_i} \cdot S_{1,i}) \cdot e(h_{2,i}^a \cdot u_{2,i}^b \cdot g_q^{\beta v_i} \cdot R_{2,i}, g_p^{r_{2,i}} \cdot g_q^{f_2 u_i} \cdot S_{2,i}) \\
&= e(g_p^a, \prod\nolimits_{i=1}^{\omega} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}) \cdot e(g_p^b, \prod\nolimits_{i=1}^{\omega} u_{1,i}^{-r_{1,i}} \cdot u_{2,i}^{-r_{2,i}}) \cdot \\
&\quad \prod\nolimits_{i=1}^{\omega} e(h_{1,i}^a \cdot u_{1,i}^b \cdot g_q^{\alpha v_i}, g_p^{r_{1,i}} \cdot g_q^{f_1 u_i}) \cdot e(h_{2,i}^a \cdot u_{2,i}^b \cdot g_q^{\beta v_i}, \\
&\quad g_p^{r_{2,i}} \cdot g_q^{f_2 u_i}) \\
&= \prod\nolimits_{i=1}^{\omega} e(g_q, g_q)^{(\alpha f_1 + \beta f_2) v_i u_i} \\
&= e(g_q, g_q)^{(\alpha f_1 + \beta f_2 mod q) \langle \vec{u}, \vec{v} \rangle},
\end{aligned}
$$

where $\langle \vec{u}, \vec{v} \rangle$ 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 $\vec{v}$ and the query vector $\vec{u}$ 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 $\vec{v} = (b_1, \cdots, b_{\frac{\omega-1}{2}}, b_{\frac{\omega+1}{2}}, \cdots, b_{\omega-1}, b_\omega) \in \{0, 1\}^\omega$, where $\{b_1, \cdots, b_{\frac{\omega-1}{2}}\}$ represents the binary code of x_H, $\{b_{\frac{\omega+1}{2}}, \cdots, b_{\omega-1}\}$ represents the binary code of y_H, and $b_\omega = 1$. If *D* is in the query range, and the query vector corresponding to *D* is $\vec{u} = (b_1^{'}, \cdots, b_{\frac{\omega-1}{2}}^{'}, b_{\frac{\omega+1}{2}}^{'}, \cdots, b_{\omega-1}^{'}, b_\omega^{'})$, 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 $b_i = b_i^{'} = 1$. In addition, $b_\omega^{'} = -\omega_1$, where ω_1 is the number of bits with the value of 1 in $(b_1^{'}, \cdots, b_{\omega-1}^{'})$, that is $\omega_1 = \sum_{i=1}^{\omega-1} b_i^{'}$ for $b_i^{'} = 1$. Thus we can get $\langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{\omega} b_i \cdot b_i^{'} = 0 \cdot (-1) + \cdots + 0 \cdot (-1) + 1 \cdot 1 + \cdots + 1 \cdot 1 - \omega_1 = 0$.

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 $\langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{\omega} b_i \cdot b_i^{'} = 0 \cdot (-1) + \cdots + 0 \cdot (-1) + 0 \cdot 1 + 1 \cdot 1 + \cdots + 1 \cdot 1 - \omega_1 \neq 0$, or $b_i^{'} = -1$ while $b_i = 1$, then $\langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{\omega} b_i \cdot b_i^{'} = 0 \cdot (-1) + \cdots + 0 \cdot (-1) + 1 \cdot (-1) + 1 \cdot 1 + \cdots + 1 \cdot 1 - \omega_1 \neq 0$.

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

When the spatial data (x, y) is outside the query range, $\langle \vec{u}, \vec{v} \rangle \neq 0$, 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 $\alpha f_1 + \beta f_2 \bmod q = 0$, then the judgment result of the *Match* algorithm is 1 and $Pr\{E = 1\} \leq negl(\lambda)$.

In summary, we can get

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

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 $\mathcal{O}(1)$ 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 *SSW.Query* algorithm, which requires $\mathcal{O}(\alpha)$, where α is the average number of query vectors contained in the atomic region. Therefore, the total time required for a range query is $\mathcal{O}(\alpha\tau)$, where τ represents the total amount of spatial data in the atomic region intersecting with the query range. However, it requires $\mathcal{O}(\alpha n)$ 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 τ 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 $\mathcal{O}(\tau')$ to update an encrypted data point (i.e., $\mathcal{O}(\tau')$ is required to find the data point to be updated, and $\mathcal{O}(1)$ is required to update the data point), where τ' 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 λ, vector space Δ_T^ω, etc., information such as data size, data structure, retrieval results are also granted to untrusted cloud servers, which are formally described as leakage function $\mathcal{L}$ and detailed as follows:

***Definition* 1.** Size Pattern (φ_1): The amount of spatial data in region *R*. For Size pattern, the input of $\mathcal{L}$ is an encrypted index Γ^* of a spatial dataset, and the output is an integer $\varphi_1 = n \leftarrow \mathcal{L}(\Gamma^*)$, where *n* represents the number of data points within the region *R*.

***Definition* 2.** Structure pattern (φ_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 $\mathcal{L}$ is the encrypted index Γ^* of the spatial dataset within a region *R*, and the output is a m'-dimensional vector $\varphi_2 = (|AR_1|, |AR_2|, \cdots, |AR_{m'}|) \leftarrow \mathcal{L}(\Gamma^*)$, 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 (φ_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 $\mathcal{L}$, where *TK* $\leftarrow TokenGen(sk, Q)$, $Q \leftarrow QueryGen(q)$, and the output of $\mathcal{L}$ is a two-dimensional vector $\varphi_3 = (|CR_1|, |CR_2|) \leftarrow \mathcal{L}(TK)$, 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 (φ_4): The number of the same atomic regions in the current query compared to the previous queries. For Search pattern, the inputs of $\mathcal{L}$ 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 $\mathcal{L}$ is a *t*-dimensional vector $\varphi_4 = (\alpha_1, \alpha_2, \cdots, \alpha_t) \leftarrow \mathcal{L}(TK, TK')$, where α_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 (φ_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 $\mathcal{L}$ are encrypted index Γ^* and token *TK*, and the output of $\mathcal{L}$ is a *n*-dimensional vector $\varphi_5 = (\beta_1, \beta_2, \cdots, \beta_n) \leftarrow \mathcal{L}(\Gamma^*, TK)$, 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-SCPA*).

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

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

Init. The adversary $\mathcal{A}$ submits two datasets DB_0 and DB_1 to challenger $\mathcal{C}$, 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 $i = 1, \cdots, n$, $D_{0,i}, D_{1,i} \in \Delta_T^\omega$ and DB_0 and DB_1 satisfy $\mathcal{L}(\Gamma_0^*) = \mathcal{L}(\Gamma_1^*)$.

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

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

- *Ciphertext Request*: For the *j*th ciphertext request, adversary $\mathcal{A}$ submits a dataset DB_j, where $DB_j = \{D_{j,1}, D_{j,2}, \cdots, D_{j,n}\}$, then challenger $\mathcal{C}$ responds with an encrypted index Γ_j^*, where $\Gamma_j^* \leftarrow DBEnc(sk, \Gamma_j)$, $\Gamma_j \leftarrow IndexGen(DB_j)$.
- *Token Request*: For the *j*th token request, adversary $\mathcal{A}$ submits a query q_j, where $\mathcal{L}(\Gamma_0^*, TK_j) =$

$\mathcal{L}(\Gamma_1^*, TK_j)$, then challenger $\mathcal{C}$ responds with a query token TK_j, where $TK_j \leftarrow TokenGen(sk, Q_j)$, $Q_j \leftarrow QueryGen(q_j)$.

Challenge. The Challenger randomly selects a bit b from $\{0, 1\}$, then invokes *IndexGen* and *DBEnc* algorithms to get index Γ_b and encrypted index Γ_b^*, and returns Γ_b^* to adversary.

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

Guess. Adversary $\mathcal{A}$ outputs a bit b' as a guess of b.

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

$$Adv_{\Pi,\mathcal{A}}^{Data}(1^\lambda) = |Pr[b' = b] - \frac{1}{2}| \leq negl(\lambda),$$

where *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 Π = (*Setup*, *IndexGen*, *DBEnc*, *QueryGen*, *TokenGen*, *Match*, *Dec*) be a symmetric key range query scheme (*ARQ*) based on the security parameter λ. We define the following security games between Challenger $\mathcal{C}$ and adversary $\mathcal{A}$:

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

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

Phase 1. Adversary $\mathcal{A}$ adaptively submits one of the following types of requests:

- *Ciphertext Request*: On the *j*th ciphertext request, the adversary $\mathcal{A}$ submits a dataset DB_j, where $DB_j = \{D_{j,1}, D_{j,2}, \cdots, D_{j,n}\}$, then Challenger $\mathcal{C}$ responds to an encrypted index Γ_j^*, where $\Gamma_j^* \leftarrow DBEnc(sk, \Gamma_j)$, $\Gamma_j \leftarrow IndexGen(DB_j)$, 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 $\mathcal{A}$ submits query q_j', where $\mathcal{L}(TK_j', TK_0) = \mathcal{L}(TK_j', TK_1)$, then challenger $\mathcal{C}$ responds with a query token TK_j', where $TK_j' \leftarrow TokenGen(sk, Q_j')$, $Q_j' \leftarrow QueryGen(q_j')$.

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

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

Guess. Adversary $\mathcal{A}$ outputs a bit b' as a guess of b.

We believe that the scheme Π is secure under Selective Chosen-Plaintext Attack if for any probability polynomial time adversary $\mathcal{A}$, the advantage of adversary $\mathcal{A}$ is a negligible function based on λ. That is:

$$Adv_{\Pi,\mathcal{A}}^{Query}(1^\lambda) = |Pr[b' = b] - \frac{1}{2}| \leq negl(\lambda),$$

where *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 Θ 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 $360°$ 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 $\mathcal{O}(2^{3l})$, 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 $\mathcal{A}$ 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^λ.

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 *IndexGen* algorithm and *DBEnc* algorithm to generate index Γ_j and encrypted index Γ_j^* respectively, and finally returns the encrypted target element $\Gamma_{j,i}^*$ as a response.

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

- *Trapdoor request*: By calling *QueryGen* algorithm, the adversary outputs the query Q_i^* and sends it to the challenger. If $\mathcal{L}(\Gamma_0^*, TK_i^*) = \mathcal{L}(\Gamma_1^*, TK_i^*)$, the challenger calls *TokenGen* algorithm to generate the token TK_i^* 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 Γ_b, and then calls the *DBEnc* algorithm to get the encrypted index Γ_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 $\mathcal{L}$ 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 $\mathcal{O}(r^2\alpha)$, where α is the average number of query vectors in an atomic region. Therefore, the computational complexity of query generation and token generation is $\mathcal{O}(r^2\alpha)$. 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 $\mathcal{O}(nr^2)$. Therefore, the calculation cost of the user is $\mathcal{O}((n+\alpha)r^2)$, 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 $\mathcal{O}(n)$. Therefore, the computation cost of LBSP is $\mathcal{O}(n)$.

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 $\mathcal{O}(n \cdot 4^{-N} + nr^2)$.

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

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 II
THE MEANING OF NOTATIONS IN PERFORMANCE ANALYSIS

Notation	Description
r	Radius of the range query
α	The average number of query vectors in an atomic region
N	Order of Hilbert curve
n	Total number of spatial data in a region
n'	Total number of spatial data in a circle range
$\|TK\|$	Size of the token
$\|D\|$	Size of the spatial data
ω	Length of location vector and query vector
$\|e_G\|$	The element size in group $\mathbb{G}$
t_p	Time cost for a bilinear mapping
t_e	Time cost for an exponential operation in $\mathbb{G}_p$
t_e'	Time cost for an exponential operation in $\mathbb{G}_q$

TABLE III
EFFICIENCY ANALYSIS

Entity	Computation Cost	Communication Cost
user	$\mathcal{O}((n+\alpha)r^2)$	$\|TK\| + n'\|D\|$
LBSP	$\mathcal{O}(n)$	$n\|D\|$
CSP	$\mathcal{O}(n \cdot 4^{-N} + nr^2)$	$\|TK\| + (n'+n)\|D\|$

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.

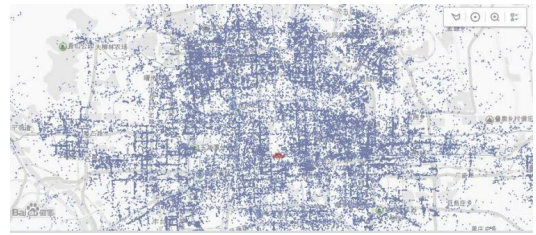


Fig. 9. Distribution of locations in the Beijing city

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 0.05*ms* to construct the Hilbert curve. Therefore, it is efficient to use Hilbert curve for region division.

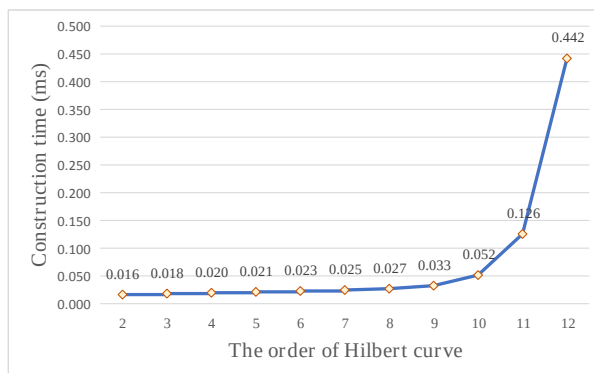


Fig. 10. Time cost for constructing Hilbert curve

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, $|e_G|$ represents the size of the elements in group $\mathbb{G}$. t_p, t_e and t_e' respectively represent the time required for a bilinear mapping, an exponential operation in group $\mathbb{G}_p$ and an exponential operation in group $\mathbb{G}_q$. Table IV describes the size of key, ciphertext and token and the computation time required for different stages in *SSW*, where ω 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

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

TABLE IV
THE PERFORMANCE OF *SSW*

Element	Size	Stage	Computation time
sk	$4(\omega+1)\|e_{\mathbb{G}}\|$	*Encrypt*	$(4\omega+2)t_e+2\omega t'_e$
CT	$2(\omega+1)\|e_{\mathbb{G}}\|$	*TokenGen*	$6\omega t_e+2\omega t'_e$
TK	$2(\omega+1)\|e_{\mathbb{G}}\|$	*Query*	$2(\omega+1)t_p$

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.

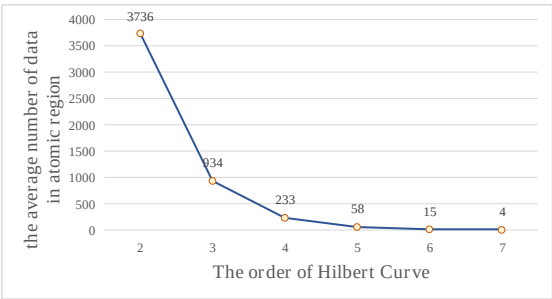


Fig. 11. The impact of Hilbert curve's order

The order N will influences the edge length d of atomic region: a larger N corresponds to a smaller d. Since the length ω of location vector and query vector is related to d, a larger N corresponds to a shorter ω. TableV 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 $1000.86m$. At this time, the length ω of location vector is 33, and it takes $823.171s$ 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 $250.215m$, and it takes $428.025s$ to encrypt the datasets. Although encrypting the spatial dataset takes some time, it is only a one-time cost.

TABLE V
IMPACT OF N ON ENCRYPTION TIME

N	d	ω	Encryption Time
1	$64055m$	33	$823.171s$
7	$1000.86m$	21	$527.260s$
8	$500.43m$	19	$477.642s$
9	$250.22m$	17	$428.025s$

Given the query size, the order N can affect the token generation time. TableVI describes the impact of N on token generation time under r = $1000m$. When N = 7, it takes about $11.592ms$ to generate a query vector and $1.669s$ to generate a token corresponding to a given query range. When $N=9$, it takes about $9.384ms$ to generate a query vector and $11.148s$ 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 VI
IMPACT OF N ON TOKEN GENERATION TIME

Query range	N	Time for a query vector	Time for token
$r=1000m$	7	$11.592ms$	$1.669s$
	8	$10.488ms$	$4.531s$
	9	$9.384ms$	$11.148s$

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 TableVII, when N = 7, there are about 10 data points in each atomic region for a given dataset, and the query time is $6.970s$. When N = 9, there are about 3 data points in each atomic region, and it takes $1.711s$ to search.

TABLE VII
IMPACT OF N ON SEARCH TIME

Query range	N	ω	Data amount in one atomic region	Search time
$r=1000m$	7	21	10	$6.970s$
	8	19	6	$3.802s$
	9	17	3	$1.711s$

The impact of N on token size is shown in TableVIII, 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 VIII
IMPACT OF N ON TOKEN SIZE

Query range	N	ω	Size of a query vector	Token size
$r=1000m$	7	21	2.178*KB*	313.632*KB*
	8	19	1.98*KB*	855.36*KB*
	9	17	1.782*KB*	2.067*MB*

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 ω = 19, we choose four different query ranges: $500m$, $1000m$ and $2000m$. The token generation time and query time are shown in TableIX.

C. Comparison with other solutions

We compare our scheme with the latest privacy-preserving range query scheme (*CRSE*[15] and *FastGeo*[7], respectively),

TABLE IX
IMPACT OF r ON TOKEN GENERATION AND SEARCH

	Query range	Token Generation Time	Search Time
$N = 8, \omega = 19$	$r = 500m$	$1.510s$	$3.802s$
	$r = 1000m$	$4.531s$	$5.619s$
	$r = 1500m$	$7.551s$	$11.651s$
	$r = 2000m$	$12.082s$	$15.206s$

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 TableX 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 is suitable for large-scale datasets and the query can be accurate to $1m$.

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.

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

[1] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[2] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE Symposium on Security and Privacy, (SP 2013)*, Berkeley, CA, USA, May, 2013, pp. 463–477.

[3] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. International Conference on the Theory and Application of Cryptographic Techniques*, Prague, Czech Republic, May, 1999, pp. 223–238.

[4] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[5] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annual ACM Symposium on Theory of Computing, (STOC 2009)*, Bethesda, MD, USA, May, 2009, pp. 169–178.

[6] Z. M. Liu, L. Wu, J. Ke, W. Qu, W. Wang, and H. Wang, "Accountable outsourcing location-based services with privacy preservation," *IEEE Access*, vol. 7, pp. 117 258–117 273, 2019.

[7] B. Wang, M. Li, and L. Xiong, "Fastgeo: Efficient geometric range queries on encrypted spatial data," *IEEE Trans. Dependable Secur. Comput.*, vol. 16, no. 2, pp. 245–258, 2019.

[8] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proc. 6th Theory of Cryptography Conference, (TCC 2009), San Francisco, CA, USA, March, 2009. Proceedings*, pp. 457–473.

[9] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory of Cryptography, (TCC 2007), Amsterdam, The Netherlands, February*, 2007, pp. 535–554.

[10] B. Wang, Y. Hou, M. Li, H. Wang, and H. Li, "Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index," in *Proc. 9th ACM Symposium on Information, Computer and Communications Security, (ASIA CCS '14), Kyoto, Japan, June, 2014*, pp. 111–122.

[11] E. Shi, J. Bethencourt, T. h. Hubert, C. Dawn, and S. A. Perrig, "Multi-dimension range query over encrypted data," in *Proc. IEEE Symposium on Security and Privacy*, 2007, pp. 350–364.

[12] P. Wang and C. V. Ravishankar, "Secure and efficient range queries on outsourced databases using rp-trees," in *Proc. 29th IEEE International Conference on Data Engineering, (ICDE 2013), Brisbane, Australia, April, 2013*, pp. 314–325.

[13] B. Wang, Y. Hou, M. Li, H. Wang, H. Li, and F. Li, "Tree-based multi-dimensional range search on encrypted data with enhanced privacy," in *Proc. International Conference on Security and Privacy in Communication Networks - 10th International ICST Conference, SecureComm 2014, Beijing, China, September, 2014, Revised Selected Papers, Part I*, pp. 374–394.

[14] Y. Lu, "Privacy-preserving logarithmic-time search on encrypted data in cloud," in *Proc. 19th Annual Network and Distributed System Security Symposium, (NDSS 2012), San Diego, California, USA, February, 2012*.

[15] B. Wang, M. Li, H. Wang, and H. Li, "Circular range search on encrypted spatial data," in *Proc. IEEE Conference on Communications and Network Security, (CNS 2015), Florence, Italy, September, 2015*, pp. 182–190.

[16] Z. Hui, R. Lu, H. Cheng, C. Le, and L. Hui, "An efficient privacy-preserving location-based services query scheme in outsourced cloud," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7729–7739, 2016.

[17] B. Wang, M. Li, and H. Wang, "Geometric range search on encrypted spatial data," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 4, pp. 704–719, 2016.

TABLE X
PERFORMANCE COMPARISON

	Scheme	Encryption	Token	Search	Complexity	Accuracy	Range type
$n = 1000, r = 100m$	*CRSE*	$5.61s$	$329.47ms$	$98.65s$	linear	$10m$	circle
	FastGeo	$19.67s$	$550.38ms$	$13.67s$	sublinear	$10m$	any shape
	Ours	$3.79s$	$377.57ms$	$3.80s$	sublinear	$1m$	any shape

[18] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[19] Y. Luo, S. Fu, D. Wang, M. Xu, and X. Jia, "Efficient and generalized geometric range search on encrypted spatial data in the cloud," in *Proc. 25th IEEE/ACM International Symposium on Quality of Service, (IWQoS 2017), Vilanova i la Geltrú, Spain, June, 2017*, pp. 1–10.

[20] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. ACM SIGMOD International Conference on Management of Data, (SIGMOD 2009), Providence, Rhode Island, USA, June, 2009*, pp. 139–152.

[21] L. Wilkinson, *The Grammar of Graphics, Second Edition*, ser. Statistics and computing. Springer, 2005.

[22] X. Li, Y. Zhu, J. Wang, and J. Zhang, "Efficient and secure multi-dimensional geometric range query over encrypted data in cloud," *J. Parallel Distributed Comput.*, vol. 131, pp. 44–54, 2019.

[23] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "k-nearest neighbor classification over semantically secure encrypted relational data," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1261–1273, 2015.

[24] L. Zhou, Y. Zhu, and A. Castiglione, "Efficient k-nn query over encrypted data in cloud with limited key-disclosure and offline data owner," *Comput. Secur.*, vol. 69, pp. 84–96, 2017.

[25] Y. Zhu, Z. Huang, and T. Takagi, "Secure and controllable k-nn query over encrypted cloud data with key confidentiality," *J. Parallel Distributed Comput.*, vol. 89, pp. 1–12, 2016.

[26] Y. Zhu, Y. Zhang, X. Li, H. Yan, and J. Li, "Improved collusion-resisting secure nearest neighbor query over encrypted data in cloud," *Concurr. Comput. Pract. Exp.*, vol. 31, no. 21, 2019.

[27] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing private queries over untrusted data cloud through privacy homomorphism," in *Proc. 27th International Conference on Data Engineering, (ICDE 2011), Hannover, Germany, April, 2011*, pp. 601–612.

[28] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE 30th International Conference on Data Engineering, Chicago, (ICDE 2014), IL, USA, March, 2014*, pp. 664–675.

[29] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. 2010 International Conference on Distributed Computing Systems, (ICDCS 2010), Genova, Italy, June, 2010*, pp. 253–262.

[30] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, 2012.

[31] Y. Yang, X. Liu, and R. H. Deng, "Multi-user multi-keyword rank search over encrypted data in arbitrary language," *IEEE Trans. Dependable Secur. Comput.*, vol. 17, no. 2, pp. 320–334, 2020.

[32] A. Akavia, D. Feldman, and H. Shaul, "Secure search on encrypted data via multi-ring sketch," in *Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security, (CCS 2018), Toronto, ON, Canada, October, 2018*, pp. 985–1001.

[33] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, "Location privacy via private proximity testing," in *Proc. Network and Distributed System Security Symposium, (NDSS 2011), San Diego, California, USA, February, 2011*.

[34] G. Zhong, I. Goldberg, and U. Hengartner, "Louis, lester and pierre: Three protocols for location privacy," in *Proc. Privacy Enhancing Technologies, 7th International Symposium, (PET 2007) Ottawa, Canada, June, 2007*, pp. 62–76.

[35] J. Sedenka and P. Gasti, "Privacy-preserving distance computation and proximity testing on earth, done right," in *Proc. 9th ACM Symposium on Information, Computer and Communications Security, (ASIA CCS '14), Kyoto, Japan, June, 2014*, pp. 99–110.

[36] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," ser. Lecture Notes in Computer Science, vol. 5126. Springer, 2008, pp. 486–498.

[37] Y. Luo, X. Jia, S. Fu, and M. Xu, "pride: Privacy-preserving ride matching over road networks for online ride-hailing service," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 7, pp. 1791–1802, 2019.

[38] A. Pham, I. Dacosta, G. Endignoux, J. R. Troncoso-Pastoriza, K. Huguenin, and J. Hubaux, "Oride: A privacy-preserving yet accountable ride-hailing service," in *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*. USENIX Association, 2017, pp. 1235–1252.

[39] Y. Guo, X. Yuan, X. Wang, C. Wang, B. Li, and X. Jia, "Enabling encrypted rich queries in distributed key-value stores," *IEEE Trans. Parallel Distributed Syst.*, vol. 30, no. 6, pp. 1283–1297, 2019.

[40] R. Poddar, T. Boelter, and R. A. Popa, "Arx: A strongly encrypted database system," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 591, 2016.

[41] H. Guo, Z. Zhang, J. Xu, N. An, and X. Lan, "Accountable proxy re-encryption for secure data sharing," *IEEE Transactions on Dependable & Secure Computing*, pp. 1–1, 2018.

[42] H. Wang, Z. Zheng, L. Wu, and P. Li, "New directly revocable attribute-based encryption scheme and its application in cloud storage environment," *Cluster Computing*, vol. 20, no. 3, pp. 2385–2392, 2017.

[43] H. Wang, D. He, J. Shen, Z. Zheng, C. Zhao, and M. Zhao, "Verifiable outsourced ciphertext-policy attribute-based encryption in cloud computing," *Soft Comput.*, vol. 21, no. 24, pp. 7325–7335, 2017.

[44] L. Li, R. Lu, and C. Huang, "EPLQ: efficient privacy-preserving location-based query over outsourced encrypted data," *IEEE Internet of Things J.*, vol. 3, no. 2, pp. 206–218, 2016.

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.

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

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.