Hindawi
Security and Communication Networks
Volume 2022, Article ID 9360899, 16 pages
https://doi.org/10.1155/2022/9360899

Research Article

Privacy-Preserving Query Scheme (PPQS) for Location-Based Services in Outsourced Cloud

Guangcan Yang,[1] **Yunhua He**,[1] **Ke Xiao**,[1] **Qifeng Tang**,[2,3] **Yang Xin**,[4] **and Hongliang Zhu**[4]

[1]*School of Information Science and Technology, North China University of Technology, Beijing 100144, China*
[2]*National Engineering Laboratory for Big Data Distribution and Exchange Technologies, Shanghai 200436, China*
[3]*Shanghai Data Exchange Corpration, Shanghai 200436, China*
[4]*School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Ke Xiao; xiaoke@ncut.edu.cn

Received 6 August 2021; Revised 14 November 2021; Accepted 6 December 2021; Published 21 January 2022

Academic Editor: Anwar Ghani

Pervasive smartphones boost the prosperity of location-based service (LBS) and the increasing data prompt LBS providers to outsource their LBS datasets to the cloud side. The privacy issues of LBS in the outsourced cloud scenario have attracted considerable interest recently. However, current schemes cannot provide sufficient privacy preservation against practical challenges and are little concerned about the data retrieval efficiency of the cloud side. Therefore, we present an efficient Privacy-Preserving LBS Query Scheme (i.e., PPQS). In our scheme, two cloud entities are employed to store the sensitive information of the outsourced data and provide the query service, which enhances the ability of privacy preservation for sensitive information. Besides, by using the techniques of homomorphic encryption and searchable symmetric encryption, the proposed scheme supports both the type query and the range query, which can significantly improve the data retrieval efficiency of the cloud side and reduce the computation burden on the cloud side and the user side. Through detailed analysis on security and computation cost, we show the enhanced ability of privacy preservation and the lower computation cost compared to previous schemes. Based on a real dataset, extensive simulations are performed to validate the effectiveness and performance of our scheme.

1. Introduction

Along with the boom of location-aware mobile electronic devices and the development of wireless communication, location-based service (LBS) has been prevalent in social domains and has attracted considerable interest recently. With the help of LBS, people can get convenience in points of interest (POI) searching, route guiding, and so forth. Nowadays, due to the advantages of cloud on data computation and storage, the location-based service provider (LBSP) tends to outsource the storage service and the query service to the cloud side [1].

A typical scenario for LBS in the outsourced cloud is shown in Figure 1. The LBSP first outsources its database that contains valuable and sensitive information (e.g., coordinates of POI) to the cloud side. Then, based on the outsourced database, the cloud side handles users' LBS queries. However, this new service paradigm brings new privacy issues since the cloud side is assumed to be semitrust (honest-but-curious). In general, the privacy issues for LBS in the outsourced cloud are two main parts: (1) Since the sensitive information contained in the outsourced database may be exposed to the cloud side, how to keep the outsourced data secret from the cloud side (i.e., how to guarantee data privacy of the outsourced data). (2) Since the private information such as the current location contained in the LBS user's query request faces the risk of being exposed to the cloud side, how to keep the private information secret from the cloud side (i.e., how to ensure the LBS user's query privacy).

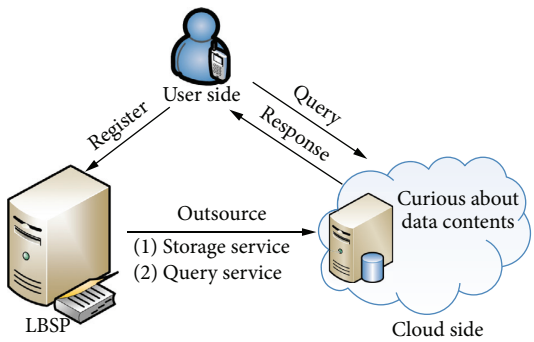

Figure 1: The privacy risks for LBS in the outsourced cloud.

To guarantee data privacy, one straightforward method is to encrypt all the data before the LBSP outsources its database to the cloud side. Similarly, to prevent the leakage of private information contained in users' queries, a common way is to encrypt the private data contained in users' queries before users submit their query requests to the cloud side. Therefore, most of the current researches adopt the way of encryption to solve the privacy issues for LBS in the outsourced cloud [2–6]. Although the above researches can preserve data privacy and query privacy, their schemes are designed based on a single cloud structure (i.e., the cloud side consists of one single cloud provider). Accordingly, both the storage service and the query service rely on a single cloud provider. However, if the single cloud provider is controlled or the data stored in this cloud are stolen by the profit-driven insider, sensitive information may be up against the crisis of being leaked since all the outsourced data can be obtained on a one-time basis. Thus, while preserving data privacy and query privacy, how to design a new scheme to enhance the ability of privacy preservation for sensitive information is an urgent problem to be solved.

Current researchers pay little attention to the data retrieval efficiency of the cloud side since their schemes are only the range query supported [2–4]. That is, a user can appoint an encrypted search area into an LBS query request and then submit this query request to the cloud side. Without knowing the user's location information, the cloud side will return all the encrypted data located in the search area as the query result to the user. Schemes that only support the range query lead to an expensive computation cost both on the cloud side and the user side. For the cloud side, it has to retrieve all the encrypted data in its database. For the user side, the user has to decrypt all the encrypted data in the query result to find the desired data. In reality, the user's query contains not only the search area but also the search interests (i.e., POI types). Thus, in addition to the range query based on the user's search area, the type query based on the user's POI query type is also an essential factor that needs to be considered for the LBS query in the outsourced cloud. Therefore, while preserving query privacy, how to design a scheme that supports both the range query and the type query to improve the data retrieval efficiency of the cloud side is a problem to be fixed.

To address the above problems, we proposed a Privacy-Preserving LBS Query Scheme (PPQS) in the outsourced cloud, which provides a stronger privacy guaranty and improves the data retrieval efficiency of the cloud side under a dual cloud structure. From the work in Ref. [7], if the sensitive knowledge is partitioned into two parts and distributed to two noncolluding clouds, the privacy can be preserved against the cloud side. Based on the idea of divide-and-conquer, the dual cloud structure consisting of two noncolluding clouds can ensure that each of the clouds only knows its own part and effectively isolates the knowledge contained in outsourced data. Therefore, two cloud entities (i.e., the type retrieval cloud and the location retrieval cloud) are adopted in our scheme to store the sensitive data (i.e., encrypted POI type data and encrypted location data) separately. Specifically, the contributions of our scheme can be summarized as below.

(1) We propose a dual cloud structure to enhance the ability of privacy preservation for sensitive information in the outsourced cloud scenario, i.e., our scheme has the ability to resist the insider attack and the eavesdropping attack while preserving data privacy and query privacy.

(2) Our scheme supports both the type query and the range query. Compared to the schemes that only support the range query, our scheme can significantly improve the data retrieval efficiency of the cloud side and reduce the computation burden on the cloud side and the user side.

The remainder of the paper is organized as follows. In Section 2, after discussing the related work, the system model, security requirements, and design goal are given. Subsequently, the basic notations and concepts are introduced in Section 3. Then, the detailed scheme, the analysis about security and computation cost, and the simulation results and corresponding analysis are given in Sections 4, 5, and 6, respectively. Finally, a conclusion is presented in Section 7.

2. Related Work

Our work focuses on the issues of privacy-preserving LBS query over outsourced encrypted data. In this section, we briefly review some related works that can be used to realize privacy-preserving LBS query.

Some early works mainly focus on the issues of privacy-preserving LBS query in the nonoutsourced cloud scenario. In this scenario, users send their query requests to the semitrust (i.e., honest-but-curious) LBSP that stores LBS data resources. To prevent the LBSP from obtaining the user's private information (e.g., the user's identity and location information), some well-known approaches like k-anonymity, dummy, spatial cloaking, and private information retrieval (PIR) are widely adopted. k-anonymity is a common way used to preserve the LBS user's private information [8], and the core of k-anonymity is to ensure that a user cannot be identified with a probability of at least $1/k$. Nevertheless, the sensitive information of users may also be leaked if k users' queries lack diversity in the sensitive attributes [9]. Dummy usually adopts the way of adding fake

users into the real user's query request to confuse the LBSP. As the LBSP cannot identify a real user from other fake users in the query process, the real user's privacy can be preserved. However, since fake users are added to the real user's query request, the communication overhead and computation cost inevitably increase [10]. To confuse the LBSP, spatial cloaking, such as transforming an LBS user's location to an obfuscation area or a cloaked area [11], is adopted to decrease the accuracy of the user's location. However, this approach achieves privacy preservation at the expense of expected locations' accuracy so that some nearby POI may be excluded [12]. PIR was first used to prevent the identifier of retrieved data from being leaked to the database server [13]. For the privacy-preserving LBS query, the user also can obtain the record from the LBSP without revealing which record he/she is interested in by using PIR [14]. For example, based on PIR, *Y* Xun et al. designed a framework to find the user's requested data without revealing to the LBSP which records are retrieved [15]. However, PIR usually brings a heavy computation cost since PIR needs a linear scan for all the data stored in the LBSP.

In the outsourced cloud scenario, current studies focus on the issues of privacy-preserving LBS query over outsourced encrypted data. For instance, to preserve data privacy of the outsourced data and the user's query privacy, a framework named FINE was designed for the privacy-preserving LBS query over outsourced encrypted data [16]. However, the framework only supports the rectangle range LBS query, which is not very practical since the user's query range is usually a circle. Subsequently, a privacy-preserving LBS query scheme was proposed to support the circle range LBS query over outsourced encrypted data [2]. However, the scheme only considers the circle range LBS query, i.e., a user can appoint only a circle into his/her LBS query and get all the encrypted data located in the circle from the cloud side. To improve the data retrieval efficiency of processing the circle range LBS query, Li et al. [3] put forward a privacy-preserving tree index structure in their scheme. To provide a flexible LBS query over outsourced encrypted data, Zhu et al. [4] designed a special polygons spatial query algorithm and proposed a privacy-preserving polygons spatial query scheme that allowed a user to appoint any polygon into an LBS query request. Afterward, to improve the query pertinence, by combining search token and inverted index technique, Zeng et al. [5] proposed a privacy-preserving generic LBS query scheme. However, the user side still faces the pressure of decrypting a matrix of n-dimensional vectors to get the desired results. Although the above studies can achieve the preservation of data privacy and query privacy in the outsourced cloud scenario, the storage service and the query service in these proposed schemes rely on a single cloud provider (i.e., using the single cloud structure), which may lead to the risk of leaking sensitive information contained in outsourced data if the single cloud is controlled or the data stored in this cloud is stolen by the profit-driven insider. Besides, most of the above schemes are only the range query supported and cannot support the type query based on users' search interests, which leads to an expensive computation cost both on the cloud side and the user side. Therefore, in this work, we use a dual cloud structure to provide a stronger privacy guaranty and design the scheme PPQS that supports both the type query and the range query to improve the data retrieval efficiency of the cloud side.

2.1. System Overview. In this section, we first describe the system model and then give the security requirements and design goal.

2.2. System Model. The system consists of four entities: Location-Based Service Provider (LBSP), Type Retrieval Cloud (TRC), Location Retrieval Cloud (LRC), and LBS User (U), as shown in Figure 2.

2.2.1. Location-Based Service Provider (LBSP). The LBSP is the owner of LBS data and is responsible for the LBS user registration. Due to the advantages of storage and computation on the cloud side, the LBSP outsources its storage service and LBS query service to the cloud side. Nonetheless, considering the value of these LBS data, the LBSP will perform some encryption operations to prevent the knowledge of data from being disclosed to the cloud side before outsourcing the LBS data to the cloud side.

2.2.2. Type Retrieval Cloud (TRC) and Location Retrieval Cloud (LRC). In our scheme, the cloud side is separated into two cloud entities: the type retrieval cloud (TRC) and the location retrieval cloud (LRC). The type retrieval cloud is responsible for storing and retrieving the encrypted type data composed of the encrypted POI type keywords, while the location retrieval cloud is in charge of storing and retrieving the encrypted location data. Note that the two cloud entities are two different cloud providers (e.g., Azure and Amazon).

2.2.3. LBS User (U). An LBS user can request an LBS query service to seek a certain POI type data within a specified range. Before issuing a query, an LBS user must be a registered user, i.e., the authenticity and legitimacy of the user should have been checked. Since the secure authentication mechanism of LBS users in the outsourced cloud has been proposed in [17], we assume that all the users are authenticated users in our scheme. However, to guarantee the LBS user's query privacy, the user will perform some encryption operations on the query content before sending the query request to the cloud side.

2.3. Security Requirements. In our scheme, the LBSP and LBS users are assumed to be honest. Specifically, the LBSP provides the LBS data accurately and LBS users perform encryption operations during the process of LBS query honestly. However, the cloud side is assumed to be honest but curious as in previous works [2, 4, 5], i.e., the type retrieval cloud and the location retrieval cloud are assumed to be honest but curious in this work. Herein, honest means each cloud entity performs protocols honestly without

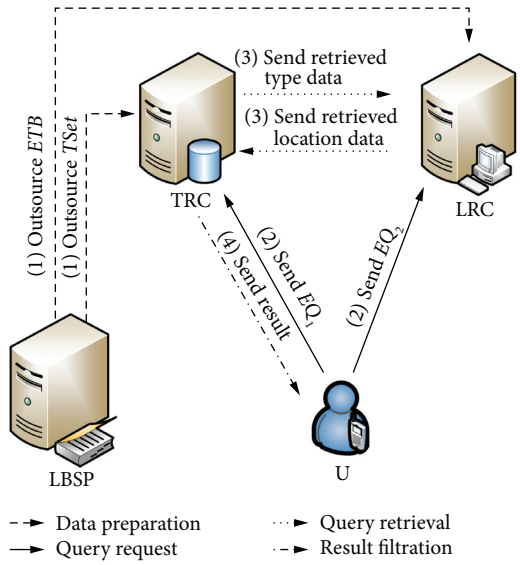

Figure 2: System model.

tampering or retaining part of data on purpose. Curious means each cloud entity is interested in the data it owns or handles, and wants to know the knowledge contained in these data. However, the type retrieval cloud and the location retrieval cloud are assumed to be two noncolluding entities in our scheme. Besides, identity privacy, the collusion attack on privacy (i.e., any two parties collude to disclose the third party's privacy), and how to prevent the above two cloud entities from collecting information from the real world to analyze the encrypted LBS data and users' queries are beyond the scope of this paper.

Under the above assumptions, to provide privacy-preserving LBS query in the outsourced cloud, the following security requirements should be satisfied in our scheme.

2.3.1. Data Privacy. The outsourced LBS data should be kept secret from the type retrieval cloud and the location retrieval cloud, i.e., our scheme should prevent the above two clouds from obtaining any actual knowledge about the outsourced data even if these data are stored in their databases.

2.3.2. Query Privacy. The knowledge contained in the user's query request should be kept secret from the type retrieval cloud and the location retrieval cloud, i.e., our scheme should prevent the above two clouds from obtaining any actual knowledge contained in the user's query request even if the above two clouds are responsible for handling the user's query and returning the query result to the user.

2.3.3. Resistance to the Insider Attack. In addition to preventing the knowledge contained in outsourced data from being leaked to the insider, our scheme should also prevent the outsourced data from being controlled or stolen on a one-time basis.

2.3.4. Resistance to the Eavesdropping Attack. In addition to preventing the knowledge contained in the user's query request from being leaked to the eavesdropper, our scheme should also prevent the knowledge contained in the user's query request from being acquired on a one-time basis.

2.4. Design Goal. Under the mentioned system model and security requirements, our design goal is to design an efficient and secure privacy-preserving LBS query scheme in the outsourced cloud scenario. The main objectives are as follows.

2.4.1. Guarantee Privacy Requirements. The proposed scheme should meet the defined security requirements. Since the type retrieval cloud and the location retrieval cloud are assumed to be honest but curious, the outsourced LBS data should be kept secret from the cloud providers otherwise the sensitive data of the LBSP could be disclosed. Similarly, the knowledge contained in the LBS user's query request should also be kept secret from the cloud entities (i.e., the type retrieval cloud and the location retrieval cloud) even if they provide the query service.

2.4.2. Perform LBS Query Efficiently. The designed scheme should achieve high time efficiency. Although the outsourced cloud provider can offer a large computing power, the data retrieval efficiency of the cloud side should be efficient for guaranteeing a short response time.

2.4.3. Achieve Low Computation Cost. Although the performance of smartphones has been greatly improved, the limitation of their batteries is still a problem. Moreover, the energy consumption of the cloud side should also be considered. Therefore, the proposed scheme should consider the computation cost for reducing the computation burden on the user side and cloud side.

3. Building Blocks

In this section, we give the notations and techniques used in this paper. The summary of notations is presented in Table 1.

3.1. Paillier Cryptosystem. Paillier cryptosystem is to solve addition operations upon the encryption field [18]. Due to additive homomorphism, the operation on encrypted data is consistent with the corresponding operation on unencrypted data. Specifically, the paillier cryptosystem consisting of three algorithms (i.e., key generation, encryption, and decryption) is shown below.

Key generation: firstly, two independent large prime numbers p and q are randomly selected. Then, we compute $n = p \cdot q$ and $\lambda = \mathrm{lcm}(p-1, q-1)$, where lcm is the least common multiple function. Finally, the public key n and private key (λ, ϕ (n)) can be obtained, where $\phi(n) = \lambda^{-1} \mathrm{mod} n$.

Encryption: assume m is a plaintext to be encrypted. Firstly, a random number $r \in \mathbb{Z}_{n^2}^*$ is selected. Then, the

Table 1: Summary of notations.

Notation	Description
(x, y)	A location coordinate
$d_{(i,j)}$	The distance from i to j
D	A data item
ID	The identifier of a data item
U	A user
W	Keywords related to POI types
DS	Description of a data item
F	A pseudo-random function, i.e., PRF
P	A pseudo-random permutation, i.e., PRP
Enc	A symmetric encryption algorithm, AES
$(\mathrm{PK}_T, \mathrm{SK}_T)$	Key pair generated by the paillier cryptosystem
$(\mathrm{PK}_U, \mathrm{SK}_U)$	A user's key pair generated by RSA algorithm
Q	An original query
Enc	A symmetric encryption algorithm, i.e., AES
H	A hash function
q_r	A query radius

encrypted result $E(m; r)$ can be computed by the following equation:

$$E(m; r)(n+1)^m \cdot r^n \bmod n^2. \tag{1}$$

Decryption: to get the plaintext m, the encrypted result $E(m; r)$ can be recovered with the private key (λ, ϕ (n)) by the following equation:

$$m = \left(\frac{\left(E(m; r)^\lambda \bmod n^2\right) - 1}{n} \right) \cdot \phi(n) \bmod n. \tag{2}$$

The property of addition homomorphic in the paillier cryptosystem can be proved by the following equation:

$$\begin{aligned} E(m_1; r_1) \oplus E(m_2; r_2) &= (n+1)^{m_1} r_1^n \cdot (n+1)^{m_2} r_2^n \\ &= (n+1)^{m_1+m_2} (r_1 \cdot r_2)^n \\ &= E(m_1 + m_2; r_1 \cdot r_2). \end{aligned} \tag{3}$$

3.2. Distance Comparison Algorithm. To guarantee the privacy of the LBS user's location and query radius, the distance comparison algorithm is proposed based on ciphertext comparison schemes [19–21], as shown in Algorithm 1. Herein, the user's query radius is denoted by q_r, and d_i represents the Euclidean distance between the user's coordinates and the coordinates of a POI, where q_r and d_i are both integers. The key generation of the paillier cryptosystem is represented by Gen, the encryption operation with public key PK_T is represented by E_{PK_T}, and the decryption operation with private key SK_T is represented by D_{SK_T}. R is the space of random number r. S is the polynomial probability algorithm, and $S(1^k, \mathrm{PK}) \subset Z$, where k is the security parameter.

4. Proposed PPQS

In this section, we give the formal descriptions of the proposed scheme, which consists of the following four phases: data preparation phase, query request phase, query retrieval phase, and result filtration phase.

4.1. Data Preparation. In general, the information of data items stored in the LBSP includes identifiers, POI type keywords, coordinates, descriptions, etc. Each data item D is plaintext in the format of $\langle \mathrm{ID}, W, (X, Y), \mathrm{DS} \rangle$, as shown in Table 2. Herein, the identifier and POI type keyword of each data item are converted to bit strings, i.e., $\mathrm{ID} \in \{0,1\}^\lambda$ and $W \subseteq \{0,1\}^*$, where λ is the security parameter of a pseudo-random function (PRF) F and a pseudo-random permutation (PRP) P.

4.1.1. Secret Key Generation. As mentioned in the system model, the LBSP outsources its encrypted LBS data to the cloud side for providing LBS users with the LBS query service. The secret key generation is the preparatory work for the outsourced storage service and query service, which mainly contains the following steps. The LBSP first chooses a secret key K_H for a secure hash function H, where H: $\{0,1\}^* \longrightarrow \mathbb{Z}_n^*$. Then, the LBSP selects a key K_S for a PRF F_1, a key K_P for a PRP P, and a key K_T for another PRF F_2. Subsequently, the LBSP selects a random number K_E for a symmetric encryption algorithm Enc (i.e., AES), where $K_E \in \mathbb{Z}_n^*$. Finally, the LBSP assigns secret keys K_H, K_S, K_T, and K_E to each registered user. Besides, the TRC runs the paillier cryptosystem to generate the key pair $(\mathrm{PK}_T, \mathrm{SK}_T)$ and then opens its public key PK_T to other entities, and the LBS user runs RSA to get the key pair $(\mathrm{PK}_U, \mathrm{SK}_U)$.

4.1.2. Encrypted Database Generation. The LBSP runs Algorithm 2 to generate two encrypted databases. Database TSet is generated by the searchable symmetric encryption (SSE) [22, 23] and contains the association between the encrypted POI-type keyword (i.e., search token stag) and the corresponding set of encrypted RID (i.e., e). Database ETB contains the relation between the encrypted ID (i.e., RID) and the corresponding encrypted coordinates (i.e., (x', y')) and encrypted description (i.e., E). In ETB, RID is generated by the pseudo-random permutation P with its key K_P, (x', y') is generated by the secure hash function H with its secret key K_H, and E is generated by the symmetric encryption algorithm Enc with its secret key K_E, as shown in Table 3. Suppose that the number of data items corresponding to the keyword $w \in W$ is denoted as $|n_w|$. Then, it needs $c_0 * |n_w|_{\max}$ times to generate the encrypted data items, where c_0 represents the number of POI-type keywords and $|n_w|_{\max}$ is the max number of data items for all the POI-type keywords. Therefore, the time complexity of Algorithm 2 is $O(c_0 |n_w|_{\max})$.

4.1.3. Query Request. The user U generates an original query request $Q = ((x, y), q_r, w)$, where (x, y) is the user's current position coordinates, q_r is the user's query radius, and w is the POI-query-type keyword.

As illustrated in the system model, the user needs to perform some encryption operations before sending the

Input: q_r, d_i.
Output: $q_r > d_i$ as TRUE or FALSE.
(1) The type retrieval cloud generates key pair $(\mathrm{PK}_T, \mathrm{SK}_T) \longleftarrow \mathrm{Gen}(1^k)$ by running the paillier cryptosystem and selects a random value $r \longleftarrow R$. The public key PK_T is open to other entities;
(2) The user encrypts query radius q_r with PK_T and then generates $\mathrm{PK}_T(q_r) \longleftarrow E_{\mathrm{PK}_T}(q_r g; r)$;
(3) The location retrieval cloud generates random $r' \longleftarrow R$ and $s \longleftarrow S$, and then computes $C_i' \longleftarrow (\mathrm{PK}_T(q_r) \cdot E_{\mathrm{PK}_T}(-(d_i + j)g; 0))^s \cdot E_{\mathrm{PK}_T}(0; r') = (\mathrm{PK}_T(q_r g; r) \cdot E_{\mathrm{PK}_T}(-(d_i + j)g); 0)^s \cdot E_{PK_T}(0; r') = \mathrm{PK}_T(s(q_r - (d_i + j))g; r^s \mathrm{o} r')$ for $j = 1, 2, \ldots n-1$, the location retrieval cloud computes C_i' and sends C_i' to the type retrieval cloud;
(4) If and only if $D_{\mathrm{SK}_T}(C_i') = 0$ is found, the type retrieval cloud outputs $q_r > d_i$ as TRUE. Otherwise outputs FALSE.

Algorithm 1: Distance comparison algorithm.

Table 2: Database form of the LBSP.

ID	W	(X, Y)	DS
$\langle 1,$	$w_1,$	(x_1, y_1)	DS
$\langle 2,$	$w_2,$	(x_2, y_2)	$\mathrm{DS}_1\rangle$
$\langle 3,$	$w_3,$	(x_3, y_3)	$\mathrm{DS}_2\rangle$
$\langle 4,$	$w_4,$	(x_4, y_4)	$\mathrm{DS}_3\rangle$
	...		
$\langle \mathrm{ID}_i,$	$w_i,$	(x_i, y_i)	$\mathrm{DS}_4\rangle$

Table 3: Database form of ETB.

RID	(x', y')	E
$\langle \mathrm{RID}_1,$	$(x_1 + H(K_H), y_1 + H(K_H))$	$E_1\rangle$
$\langle \mathrm{RID}_2,$	$(x_1 + H(K_H), y_2 + H(K_H))$	$E_2\rangle$
$\langle \mathrm{RID}_3,$	$(x_1 + H(K_H), y_3 + H(K_H))$	$E_3\rangle$
$\langle \mathrm{RID}_4,$	$(x_1 + H(K_H), y_4 + H(K_H))$	$E_3\rangle$
	...	
$\langle \mathrm{RID}_i,$	$(x_1 + H(K_H), y_i + H(K_H))$	$E_i\rangle$

Input: LBSP, K_S, K_P, K_T, K_E, K_H,
Output: TSet ETB,
(1) Initialize T to an empty array, ETB to an empty database, and t to an empty list;
(2) **for** each $w \in W$ **do**
(3) $c \longleftarrow 0$
(4) $s \longleftarrow F_1(K_S, w)$, and $\mathrm{stag} \longleftarrow F_2(K_T, w)$;
(5) **for** each $\mathrm{ID} \in \mathrm{DB}(w)$ **do**
(6) $\mathrm{RID} \longleftarrow P(K_P, \mathrm{ID})$;
(7) $(x', y') \longleftarrow (x + H(K_H), y + H(K_H))$;
(8) $E \longleftarrow \mathrm{Enc}(K_E, \mathrm{DS})$;
(9) $\mathrm{ETB} \longleftarrow (\mathrm{RID}, (x, y), E)$;
(10) $l \longleftarrow F_2(\mathrm{stag}, c)$
(11) $e \longleftarrow \mathrm{Enc}(s, \mathrm{RID})$;
(12) $t \longleftarrow e$;
(13) **end for**
(14) $T \longleftarrow (l, t)$;
(15) $\mathrm{TSet} \longleftarrow T$;
(16) $c_0 \longleftarrow c + 1$;
(17) **end for**

Algorithm 2: Encrypted database generation.

original query for guaranteeing query privacy. More specifically, the user uses the POI-type keywords w and K_T to generate search token stag, generates secret key s with w and K_S, uses K_H and the hash function H to encrypt the current coordinates, and encrypts the query radius q_r with public key PK_T. Finally, the user sends the query request EQ_1 to the type retrieval cloud and sends query request EQ_2 to the location retrieval cloud. The pseudocode of query request is shown as Algorithm 3.

4.2. Query Retrieval. The query retrieval consists of two processes: the type retrieval and the range retrieval. In brief, the type retrieval is to find the matched data that correspond to the user's POI query type and the range retrieval is to perform distance calculation on the ciphertext domain for the matched data.

4.2.1. Type Retrieval. After receiving the user's query request EQ_1, the TRC can find out a list t from TSet according to the user's stag. In list t, the POI type of each e (i.e., encrypted RID) is consistent with the user's POI query type keyword w. Afterward, the TRC inserts all the e contained in list t into an empty list t_a and then sends t_a to the LRC. The pseudocode of type retrieval process is shown as Algorithm 4. According to the user's search token stag, the TRC requires at most $O(c_0)$ times to find out list t. Therefore, the time complexity of Algorithm 4 is $O(c_0)$ at most.

4.2.2. Range Retrieval. After receiving the user's query request EQ_2, the LRC first gets the set of RID by decrypting e in list t_a with s. Then, according to the set of RID, the LRC finds out the corresponding encrypted coordinates (x', y') and encrypted description E in ETB. For each RID_i, the LRC performs the following steps: calculates the Euclidean distance between (x_i', y_i') and (x'', y'') (i.e., $d_i = \mathrm{dis}((x_i', y_i'), (x'', y''))$), runs the distance comparison algorithm and calculates C_i', generates e_i' by encrypting E_i with the user's public key PK_U, and inserts C_i' and e_i' to an empty list t_b. The pseudocode of range retrieval process is shown as Algorithm 5. Since the number of RID is consistent with the number of data that correspond to the user's POI query type keyword w, the times of decrypting e is $|n_w|$. Besides, since the LRC needs to run the distance comparison algorithm once for each RID, the corresponding times of running the distance comparison algorithm is $|n_w|$. Therefore, the time complexity of Algorithm 5 $O(|n_w|_{\max^2})$ is at most.

4.3. Result Filtration. Based on the list t_b, the TRC figures out the data that locate in the user's query radius by decrypting C_i' with the private key SK_T. If $D_{\mathrm{SK}_T}(C_i') = 0$, the TRC inserts the corresponding e_i' into list and sends as the query result to the user. The pseudocode of result filtration is shown in Algorithm 6. Since the TRC needs to decrypt all C' in list t_b to get list and the number of C' is consistent with the number of RID, the times of decrypting C' is $|n_w|$. Therefore, the time complexity of Algorithm 6 is $O(|n_w|_{\max})$ at most.

After getting res, the user can obtain the desired data by decrypting res with the private key SK_U and K_E.

5. Discussion

In this section, the security analysis is first presented to check whether the defined security requirements can be satisfied in our scheme. Subsequently, in terms of the number of data that need to be processed in different phases, the computation cost of the previous schemes and our scheme are analyzed and compared.

5.1. Security Analysis. Security analysis is based on the proposed security requirements: data privacy, query privacy, resistance to the insider attack, and resistance to the eavesdropping attack. Before analyzing the security requirements, the following lemmas are introduced to show the security of pseudo-random permutation PRP, pseudo-random permutation PRF, symmetric encryption algorithm AES, and the paillier cryptosystem.

Lemma 1 (see [5]). *For an adversary A who uses probabilistic polynomial time (PPT) algorithm, if its advantages* $\mathbf{a}\ \mathbf{dv}_{F,A}^{PRF}(\lambda)$ *and* $\mathbf{a}\ \mathbf{dv}_{F,A}^{PRP}(\lambda)$ *are negligible, then the PRF and PRP are secure, where*

$$\begin{aligned} \mathrm{adv}_{F,A}^{\mathrm{PRF}}(\lambda) &= \Pr\left[A^{F(K,\cdot)}\left(1^{\lambda}\right) = 1\right] - \Pr\left[A^{f(\cdot)}\left(1^{K}\right) = 1\right] \text{and} \\ \mathrm{adv}_{F,A}^{\mathrm{PRP}}(\lambda) &= \Pr\left[A^{F(K,\cdot)}\left(1^{\lambda}\right) = 1\right] - \Pr\left[A^{f(\cdot)}\left(1^{K}\right) = 1\right]. \end{aligned} \tag{4}$$

Input: K_S, K_H, K_T, PK_T, SK_U,
Output: EQ_1, EQ_2,
(1) U initializes an original query $Q = ((x, y), q_r, w)$;
(2) $\mathrm{stag} \longleftarrow F_2(K_T, w)$, $s \longleftarrow F_1(K_S, w)$;
(3) $(x'', y'') \longleftarrow (x + H(K_H), y + H(K_H))$;
(4) $\mathrm{PK}_T(q_r) \longleftarrow E_{\mathrm{PK}_T}(q_r g; r)$;
(5) U sends $\mathrm{EQ}_1 = \mathrm{stag}$ to the TRC;
(6) U sends $\mathrm{EQ}_2 = \langle s \| (x'', y'') \| \| \mathrm{PK}_T(q_r) \| \mathrm{PK}_U \rangle$ to the LRC;

Algorithm 3: Query request.

Input: EQ_1, TSet,
Output: t_a
(1) Initialize t_a to an empty list;
(2) $c \longleftarrow 0$;
(3) **while** $c < c_0$ **do**
(4) $l \longleftarrow F_2(\mathrm{stag}, c)$;
(5) **if** $l \in$ TSet **then**
(6) $t \longleftarrow \mathrm{TSet}[l]$;
(7) $t_a \longleftarrow t$;
(8) **else**
(9) $c \longleftarrow c + 1$;
(10) **end if**
(11) **end while**
(12) the TRC sends t_a to the LRC;

Algorithm 4: Type retrieval.

Input: ETB, EQ_2
Output: t_b
(1) Initialize to an empty list;
(2) **for** each e in t_a**do**
(3) $\mathrm{RID} \longleftarrow \mathrm{Dec}(s, e)$;
(4) **for** each RID_i**do**
(5) Calculates $d_i = \mathrm{dis}((x'_i, y'_i), (x'', y''))$;
(6) Performs the distance comparison algorithm and calculatesC'_i;
(7) $e'_i \longleftarrow \mathrm{PK}_U(E_i)$;
(8) **end for**
(9) $t_b \longleftarrow (C'_i, e'_i)$;
(10) **end for**
(11) the LRC sends t_b to the TRC;

Algorithm 5: Range retrieval.

Lemma 2 (see [6]). *For an adversary A who uses probabilistic polynomial time (PPT) algorithm, if its advantages* $\mathbf{adv}_{\Sigma,A}(\lambda)$ *is negligible, then the symmetric encryption scheme* $\Sigma = (Enc, Dec)$ *is secure, where*

$$\mathrm{adv}_{\Sigma,A}(\lambda) = \Pr\left[A^{(K,0,\cdot,\cdot)}\left(1^{\lambda}\right) = 1\right] - \Pr\left[A^{(K,1,\cdot,\cdot)}\left(1^{k}\right) = 1\right]. \quad (5)$$

Lemma 3 (see [24]). *For an adversary A who uses probabilistic polynomial time (PPT) algorithm, if its advantages* $\mathbf{adv}_{DRC,A}(\varepsilon)$ *is negligible, then the paillier cryptosystem is a* $(N, 2, e)$*-decisional composite residuosity (DRC) problem and secure, where*

$$\Pr\left[\mathscr{A}(N, 2, e, x_0) = 1\right] - \Pr\left[\mathscr{A}(N, 2, e, x_1) = 1\right] \leq \mathbf{adv}_{\mathrm{DRC},A}(\varepsilon). \quad (6)$$

5.1.1. Data Privacy

Theorem 1. *Based on the security of PRP, PRF, symmetric encryption algorithm AES, hash function H, and the paillier cryptosystem, our scheme can achieve data privacy.*

Proof. In our scheme, two encrypted databases (i.e. TSet, ETB) are outsourced to the TRC and the LRC, respectively.

Input: t_b, SK_T
Output: res
(1) Initialize to an empty list;
(2) **for** each C_i' in t_b **do**
(3) **if** $D_{SK_T}(C_i') = 0$ **then**
(4) res $\longleftarrow e_i'$;
(5) **end if**
(6) **end for**
(7) the TRC sends res to the user U;

Algorithm 6: Result filtration.

In TSet, the relation between the POI-type keyword w and encrypted index l is built by search token stag, where l and are both generated by the pseudo-random function F_2. Moreover, the ciphertext e is generated by the symmetric encryption algorithm AES with its secret key and an input RID, where s is the outcome produced by the pseudo-random function F_1 and RID is an encrypted outcome generated by the pseudo-random permutation P. Therefore, if the PRP, PRF, and symmetric encryption are secure, the TRC cannot obtain knowledge from TSet even if the TRC owns database TSet. In ETB, the encrypted ID (i.e. RID), encrypted coordinates (i.e. (x', y')), and the encrypted description (i.e. E) are generated by the pseudo-random permutation P, the hash function H, and the symmetric encryption algorithm AES, respectively. Thus, as long as the PRP, hash function, and symmetric encryption are secure, the LRC cannot get any actual knowledge from ETB even if the LRC owns database ETB.

Besides, the retrieved data e contained in list t_a sent from the TRC to the LRC is generated by the symmetric encryption AES. Similarly, the retrieved data C' contained in list t_b sent from the LRC to the TRC is produced by the paillier cryptosystem. Therefore, if the symmetric encryption and the paillier cryptosystem are secure, the knowledge contained in transferred data between the two clouds (i.e., the TRC and the LRC) can also be well protected. Thus, under the assumption that the two clouds are two noncolluding entities, no single cloud can obtain the knowledge of the data stored in itself unless the other cloud provides additional information (i.e., the two clouds collude with each other and share the secret key s and SK_T).

Therefore, our scheme can provide data privacy. □

5.1.2. Query Privacy

Theorem 2. *Based on the security of PRP, PRF, symmetric encryption algorithm, the hash function, and the paillier cryptosystem, our scheme can guarantee the user's query privacy.*

Proof. In the user's query request EQ_1, the user's POI query type keyword is represented by search token stag, where is generated by the pseudo-random function F_2. During the process of type retrieval, the stag is converted to the encrypted index l for finding the ciphertext e in list t_a, where l is generated by the pseudo-random function F_2 with stag, e is generated by the symmetric encryption algorithm AES with its secret key s and an input RID, s is the outcome produced by the pseudo-random function RID, and is an encrypted outcome generated by the pseudo-random permutation P. Therefore, although the process of type retrieval is performed on the TRC, the TRC cannot learn any useful knowledge about the user's query type due to the security of PRP, PRF, and symmetric encryption. In the user's query request EQ_2, the user's coordinates and query radius are encrypted by the hash function H and the paillier cryptosystem, respectively. During the process of range retrieval, the distance comparison between the user's query radius and the Euclidean distance that indicates the user's coordinates and the coordinates of a POI is performed on the ciphertext domain. Therefore, if the hash function and the paillier cryptosystem are secure, the LRC cannot obtain any actual knowledge about the user's accurate location and query radius.

Therefore, our scheme can provide query privacy. □

5.1.3. Resistance to the Insider Attack

Theorem 3. *Based on Theorem 1 and the proposed dual cloud structure, our scheme has the ability to resist the insider attack.*

Proof. As mentioned in Theorem 1, data privacy can be achieved. Thus, our scheme can prevent the knowledge stored in outsourced data from being leaked to the cloud service provider including its insider. However, under the single cloud structure, the outsourced data stored in the cloud side may be controlled or stolen by the insider on a one-time basis since there is only one cloud service provider (i.e., only one cloud entity). Nevertheless, the cloud side in our scheme is divided into two noncolluding cloud entities (i.e., the TRC and the LRC) and the sensitive information (i.e., POI-type information and POI location information) in the outsourced data is stored in the above two cloud providers separately. Therefore, unless the insider occupies the two cloud entities simultaneously, our scheme can prevent the insider from controlling or stealing all the outsourced data on a one-time basis.

Thus, compared to the schemes under the single cloud structure, our scheme provides a stronger ability of privacy preservation for sensitive information contained in the outsourced data. □

5.1.4. Resistance to the Eavesdropping Attack

Theorem 4. *Based on Theorem 2 and the proposed dual cloud structure, our scheme has the ability to resist the eavesdropping attack.*

Proof. As mentioned in Theorem 2, the user's query privacy can be achieved. Thus, our scheme can prevent the knowledge contained in the query request from being leaked to the eavesdropper. However, under the single cloud structure, all the user's query private data may be obtained by the eavesdropper on a one-time basis since these data are contained in one query request. Nevertheless, the user's query private data (i.e., POI-type keyword, location, and query radius) in our scheme are sent to the TRC and the LRC by two separate query requests (i.e. EQ_1, and EQ_2). Therefore, unless the eavesdropper captures the two query requests simultaneously, our scheme can prevent the eavesdropper from obtaining all the user's query private data on a one-time basis.

Therefore, compared to the schemes under the single cloud structure, our scheme provides a stronger ability of privacy preservation for sensitive information contained in the user's query request. □

5.2. Cost Analysis. Herein, we compare our scheme with previous schemes by analyzing the linear relationship between the computation cost and the number of data that need to be processed in different phases. The notations in this section are described in Table 4 and the comparison results of computation cost with the previous schemes [2, 4, 5] are concluded and shown in Table 5.

The encrypted database is generated by the LBSP, so the phase of the encrypted database generation reflects the computation cost of the LBSP. In the phase of encrypted database generation, the computation cost of each scheme in Table 5 is linear to N.

The query retrieval is performed by the cloud side, so the phase of the query retrieval reflects the computation cost of the cloud side. In the phase of query retrieval, the computation cost of scheme EPQ and scheme are linear to N. However, the computation cost of scheme P^3GQ is linear to 2 η and the computation cost of our scheme is linear to η. The reason is that the scheme EPQ and scheme Polaris are both only the range query supported, so all the data items need to be compared to decide whether these data items are within the user's query range. Therefore, the computation cost of scheme EPQ and scheme Polaris are linear to N. However, the scheme P^3GQ and our scheme P^3GQ support both the type query and the range query. Therefore, in scheme P^3GQ and our scheme, the first step of the query retrieval phase is to find the data items that match the user's POI query type keyword, and then only the matched data items need to be compared to decide whether they are within the user's query range.η is less than N since η is the number of data items for one kind of POI type. Therefore, in the phase of query retrieval, the computation cost of our scheme is less than other schemes.

The result filtration is to prepare an encrypted dataset (i.e., the query result) in which each encrypted data satisfy the user's query conditions, so the number of data in the encrypted dataset reflects the computation cost of the user side. In the phase of result filtration, the computation cost of scheme EPQ and scheme are linear to ζ, where ζ also represents the number of data located in the user's query range after performing the range retrieval. However, the computation cost of scheme P^3GQ and our scheme is linear to ξ, where ξ also denotes the number of data that not only correspond to the user's query type but locate in the user's query range after performing the type retrieval ξ and the range retrieval. is less than ζ since there is not only one kind of POI type data in the user's query range. Therefore, in the phase of result filtration, the computation cost of our scheme is the same as that of scheme P^3GQ, but less than that of scheme EPQ and scheme Polaris.

6. Evaluation

In this section, we first describe the simulation setup and then give the simulation results and corresponding analysis.

6.1. Setup. Implementations: our scheme is implemented with *Python* programming language and conducted on a Windows machine with an Intel Core-i7 3.6 GHz, 16 GB RAM, and Microsoft Windows 7 OS. Besides, we call the encrypted sub-package in the Pycryto encryption library to implement the encryption algorithms of our scheme and adopt a 512 -bit paillier cryptosystem to encrypt the coordinates of each data item. Moreover, the LBS resources are collected from an open map of Beijing by using the public API interface of the Amap service [25] and the database of the LBSP is built in the form of Table 2. Based on the database of LBSP, we construct a dataset DS with 10000 data items and the coordinates of these data items are randomly distributed in 10 km × 10 km square area LA. Besides, the encrypted database in the cloud side is implemented by constructing the type index table and the location data table in MYSQL, so the delay time of data transmission between clouds is assumed to be 0. To evaluate the time cost, retrieval efficiency, and computation burden, the simulations are performed in the following two scenarios.

Scenario 1: based on the dataset DS, three original datasets (i.e.ODS_1,ODS_2, ODS_3, and) containing two assigned POI-type data (i.e., the catering service denoted $Type_1$ as and the accommodation service denoted $Type_2$ as) are formed. Specifically, the number of $Type_1$ and $Type_2$ in original dataset ODS_1 are 1000 and 500, respectively, the number of $Type_1$ and $Type_2$ in original dataset ODS_2 are 2000 and 1000, respectively, the number of $Type_1$ and $Type_2$ in original dataset ODS_3 are 3000 and 1500, respectively. Moreover, the total number of data items in each original dataset is 10000. This scenario is used to evaluate the time cost of generating outsourced datasets.

Scenario 2: to simulate the data retrieval service, two POI query types (i.e. $Type_1$, and $Type_2$), 5 randomly selected locations in LA, and 5 specified query radii (i.e., ranging

Table 4: Notations.

Notation	Description
N	The total number of LBS data items
H	One hash function operation
SE	One symmetric encryption operation
SD	One symmetric decryption operation
EX	One exponentiation operation in group G
P	One pairing operation in group G
AE	One asymmetric encryption operation
AD	One asymmetric decryption operation
PA	One additive homomorphism operation in the paillier cryptosystem
F'	One operation of pseudo-random permutation
F''	One operation of pseudo-random function
M_1	One multiplication operation in group G
M_2	One multiplication operation between a $n \times n$ matrix and a $n$dimensional vector
M_3	One inner-product operation of two$n$-dimensional vectors
τ	The number of edges of a polygon
η	The number of data after type retrieval
ς	The number of data after range retrieval
ξ	The number of data after type retrieval and range retrieval

Table 5: Computation cost comparison with previous schemes.

Scheme	Data preparation	Query process		
	Encrypted database generation	Query request	Query retrieval	Result filtration
EPO [2]	$N(2(H + 1M_1 + 2\text{EX}) + S)$	$2H + 4\text{EX} + 4\text{AE}$	$N(2P + 3M_1 + 4)$	$\xi(\text{SE})$
Polaris [4]	$N(3H + \text{SE} + 2\text{EX})$	$\tau(H + M_1\_\text{EX} + :$	$N(4\tau(P + M_1))$	$\xi(\text{AE})$
P^3GQ [5]	$N(F' + 3F'' + \text{EX} + 2M_2)$	$2F'' + \text{EX} + 2M_2$	$2\eta(\text{SD} + \text{EX} + M_2)$	$\xi(\text{SE})$
PPQS (our scheme)	$N(F' + 2F'' + 2\text{SE} + 2H)$	$2(F'' + H) + \text{PA}$	$F'' + \eta(\text{SD} + \text{PA}$	$\xi(\text{AE})$

from 0.5 km to 2.5 km with step length 0.5 km) are assigned into the LBS query request. In addition, the grid structure with the grid cell size of 1 km × is constructed for LA, and the method of related area list [2] is used to define the LBS user's query area. This scenario is used to evaluate the time cost of the LBS query request generation, the data retrieval efficiency of the outsourced cloud, and the computation burden on the user side and cloud side.

6.2. Simulation Results

6.2.1. Time Cost of Outsourced Datasets Generation. For evaluating the time cost of generating outsourced datasets, each of the three original datasets (i.e., ODS_1, ODS_2, and ODS_3) is executed 10 times to generate the corresponding encrypted data items. Figure 3 shows the average time cost of generating outsourced datasets (i.e., DS_1, DS_2, and DS_3).

As can be seen from Figure 3, the time cost of generating outsourced datasets is increasing with the number of the assigned POI-type data in original datasets. To generate encrypted data items, the nonassigned POI-type data only need to hash their coordinates, while the assigned POI-type data not only need to hash their coordinates but also need to construct searchable encrypted indexes. Therefore, the increasing number of assigned POI-type data increases the time cost of generating outsourced datasets.

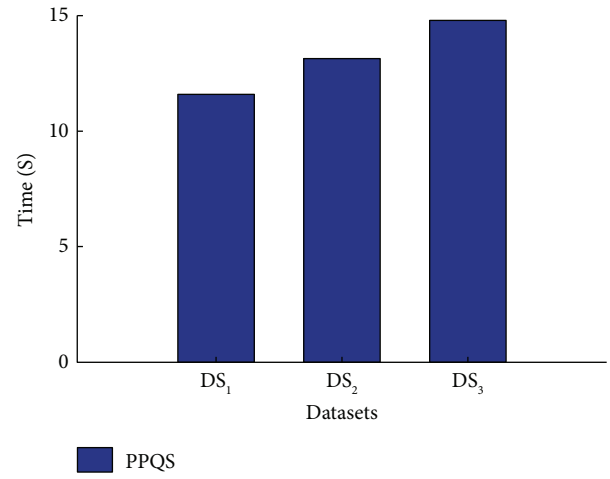

Figure 3: Time cost of outsourced datasets generation.

6.2.2. Time Cost of the LBS Query Generation. The efficiency of generating an LBS query request is important to the user side. Therefore, to evaluate the time cost of the LBS query request generation, 5 random locations in square area are selected as the LBS user's initial coordinates based on the setting of scenario 2, and each location is performed 10 times for generating the LBS query request. The average time cost of generating the LBS query request at each location is calculated and Figure 4 shows the average time cost of

generating the encrypted LBS query request at different locations.

As can be seen from Figure 4, when generating the LBS query request at different locations, the time cost of generating the encrypted LBS query request is stable, and it has little relationship with the locations. The reason is that no matter which location is selected as the user's current coordinates to generate the encrypted LBS query request, the way of encryption on the coordinates is the same (i.e., using the hash function to encrypt the user's current coordinates). In addition, the time cost of the LBS query request generation has little to do with the POI types contained in the LBS query request. The reason is that the user's POI query types are represented by the corresponding POI type keywords that are finally converted to search tokens by the pseudo-random function.

6.2.3. Retrieval Efficiency of the Outsourced Cloud. The data retrieval time is related to the data retrieval efficiency of the outsourced cloud. Therefore, to evaluate the data retrieval efficiency of the outsourced cloud, the data retrieval service is performed 10 times on each outsourced dataset (i.e. DS_1, DS_2, and DS_3) based on the setting of scenario 2. Figure 5 shows the average time cost of data retrieval service with different POI query types and query radii under different outsourced datasets.

Based on Figure 5, we can conclude how the time cost of data retrieval service is affected by the user's query radius, the data density of POI query type, the dataset density of POI type, and the supported query mode.

To research the relation between the time cost and the user's query radius, the user's query radius is selected from 0.5 km to 2.5 km. When the POI query type is fixed (e.g.,$Type_1$), the time cost of data retrieval service is increasing with the query radius no matter on which outsourced dataset. The reason is that the increase of the user's query radius leads to the enlargement of the user's query area, which leads to the increase of the data located in the user's query area. Accordingly, the cloud side needs to run more distance comparison operations, which finally leads to the increase of the time cost of data retrieval service.

Since the data amount of is twice that of $type_2$ in each outsourced dataset, the data density of $type_1$ can be seen as a high-density POI type if the data density of is $type_2$ assumed to be a low-density POI type. From Figure 5, it can be seen that the time cost of data retrieval service for the high-density POI type (i.e. $type_1$) is higher than that for the low-density POI type (i.e.$Type_2$) no matter on which outsourced dataset. The reason is that the user's query radius decides the query area, and when the query area is fixed, the amount of data of the high-density POI type is greater than that of the low-density POI type in the query area. Accordingly, a larger amount of data requires more distance comparison operations to be run, which leads to the increase of time cost. Besides, from Figure 5, the time cost on these two POI types in each outsourced dataset has similar behavior. The reason is that the time cost on the cloud side is basically spent on the distance comparison of the assigned POI-type data located in the query area during the data retrieval service. When the data amount of these two POI types in each outsourced dataset is proportional (i.e., the data amount of $type_1$ is twice that of $type_2$ in each outsourced dataset), the corresponding data amount of these two POI types in the query area is proportional. Accordingly, the time cost of data retrieval service on these two POI types (i.e., $Type_1$and$Type_2$) is proportional in each outsourced dataset, which leads to similar behavior of the time cost on these two POI types in each outsourced dataset.

Recall that the data amount of any POI type (no matter $type_1$ or $type_2$) in DS_3 is more than that in DS_1 and DS_2. Thus, in terms of the dataset density of POI type, DS_3 can be seen as a high-density outsourced dataset if DS_1 or DS_2 is assumed to be a low-density outsourced dataset. From Figure 5, for any POI type (e.g., $Type_1$), it can be seen that the time cost of data retrieval service on a high-density outsourced dataset (e.g., DS_3) is greater than that on a low-density outsourced dataset (e.g. DS_1,). The reason is that when the query area is fixed, the amount of POI-type data contained in a high-density outsourced dataset is more than that contained in a low-density outsourced dataset, i.e., the running times of distance comparison operation on a high-density outsourced dataset are more than that on a low-density outsourced dataset in the query area, which leads to a greater time cost of data retrieval service on a high-density outsourced dataset.

Figure 5 reflects the results about the time cost of data retrieval service with the assigned POI types (i.e., the results are obtained based on the query mode that supports both the type query and the range query), which means the cloud side only needs to run distance comparison operation for the matched POI-type data in the query area. However, if the type query is not supported, the cloud side needs to run the distance comparison for all the data rather than the matched POI-type data in the query area, which will undoubtedly lead to a bigger computation burden and a greater time cost of data retrieval service. Therefore, compared to the scheme that only supports the range query, the scheme (i.e. PPQS) that supports both the type query and the range query can effectively reduce the computation burden on the cloud side and improve the data retrieval efficiency of the cloud side.

6.2.4. Computation Burden on the User Side. The amount of encrypted data in the query result returned to the user from the cloud side is related to the computation burden on the user side. Although different schemes use diverse decryption algorithms to enable the user to get the desired result by decrypting the encrypted data in the query result, if the user side is assumed to adopt the same decryption algorithm, the number of encrypted data contained in the query result returned to the user can be used as the basis for measuring the user's computation burden. Therefore, the computation burden on the user side is evaluated according to the number of data in the query result. Besides, since our scheme provides the mode that supports both the type query and the range query (i.e. $T\&R$), the data retrieval service is first executed 10 times on each outsourced dataset (i.e., DS_1, DS_2,

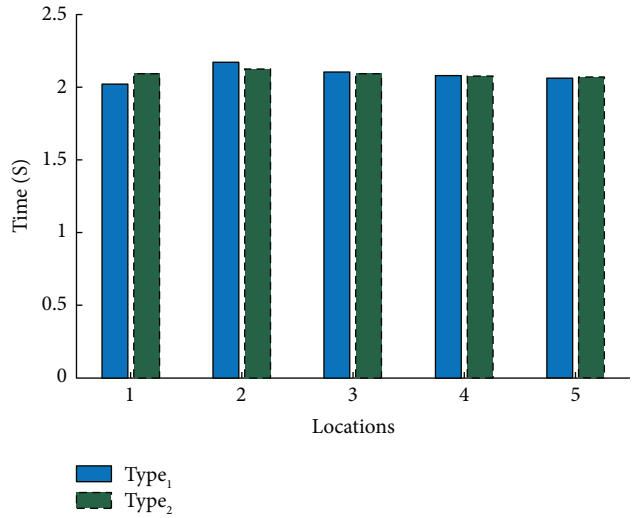

Figure 4: Time cost of the LBS query request generation.

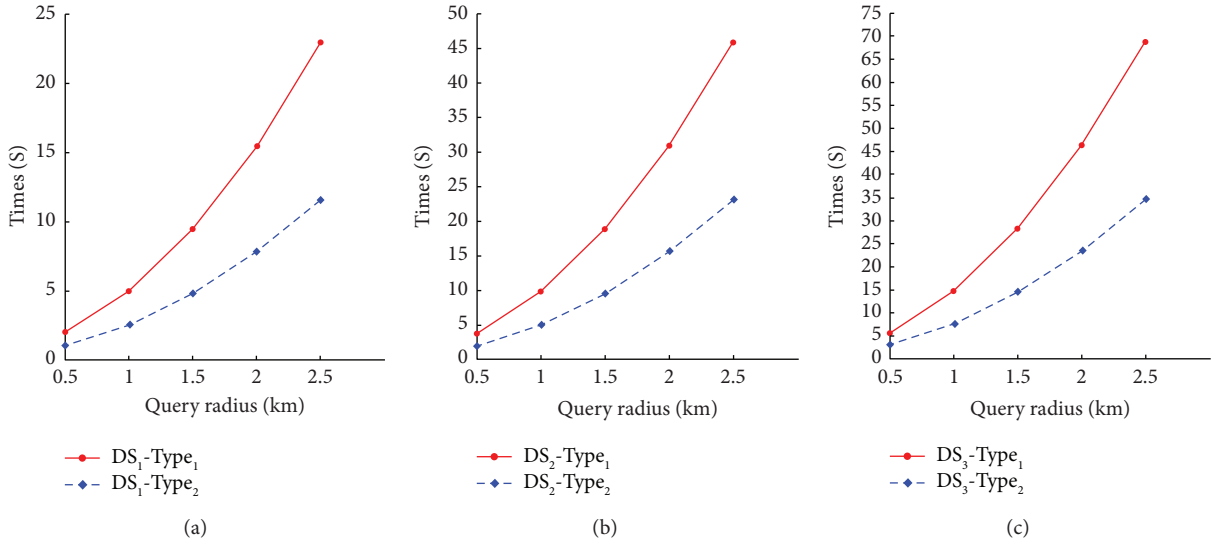

Figure 5: Time cost of data retrieval service on different outsourced datasets. (a) Time cost of data retrieval service on outsourced dataset DS_1. (b) Time cost of data retrieval service on outsourced dataset DS_2. (c) Time cost of data retrieval service on outsourced dataset DS_3.

and DS_3) based on the setting of scenario 2. To compare with the mode that only supports the range query (i.e. R), under the condition of removing the type index table, the data retrieval service is then executed 10 times on each outsourced dataset. Finally, the average number of data contained in the query result returned to the user side is calculated and Figure 6 shows the average number of data contained in the query result with different POI query types and query radii under different outsourced datasets.

Based on Figure 6, we can conclude how the number of data contained in the query result is affected by the user's query radius, the data density of POI query type, the dataset density of POI type, and the supported query mode.

When the POI query type is fixed (e.g., $Type_1$), the number of data contained in the query result is increasing with the user's query radius no matter on which outsourced dataset. The reason is that the increase of the query radius leads to the enlargement of the user's query area, which leads to the increase of the data located in the user's query area. Accordingly, the cloud side needs to insert more data into the query result.

Recall that the data density of $type_1$ can be seen as a high-density POI type if the data density of $type_2$ is assumed to be a low-density POI type. In the mode that supports both the type query and the range query (i.e., $T\&R$), it can be seen that the number of data contained in the query result for the high-density POI type (i.e., $Type_1$) is greater than that for the low-density POI type (i.e., $Type_2$) no matter on which outsourced dataset. The reason is that when the query area is fixed, the amount of the high-density POI-type data is

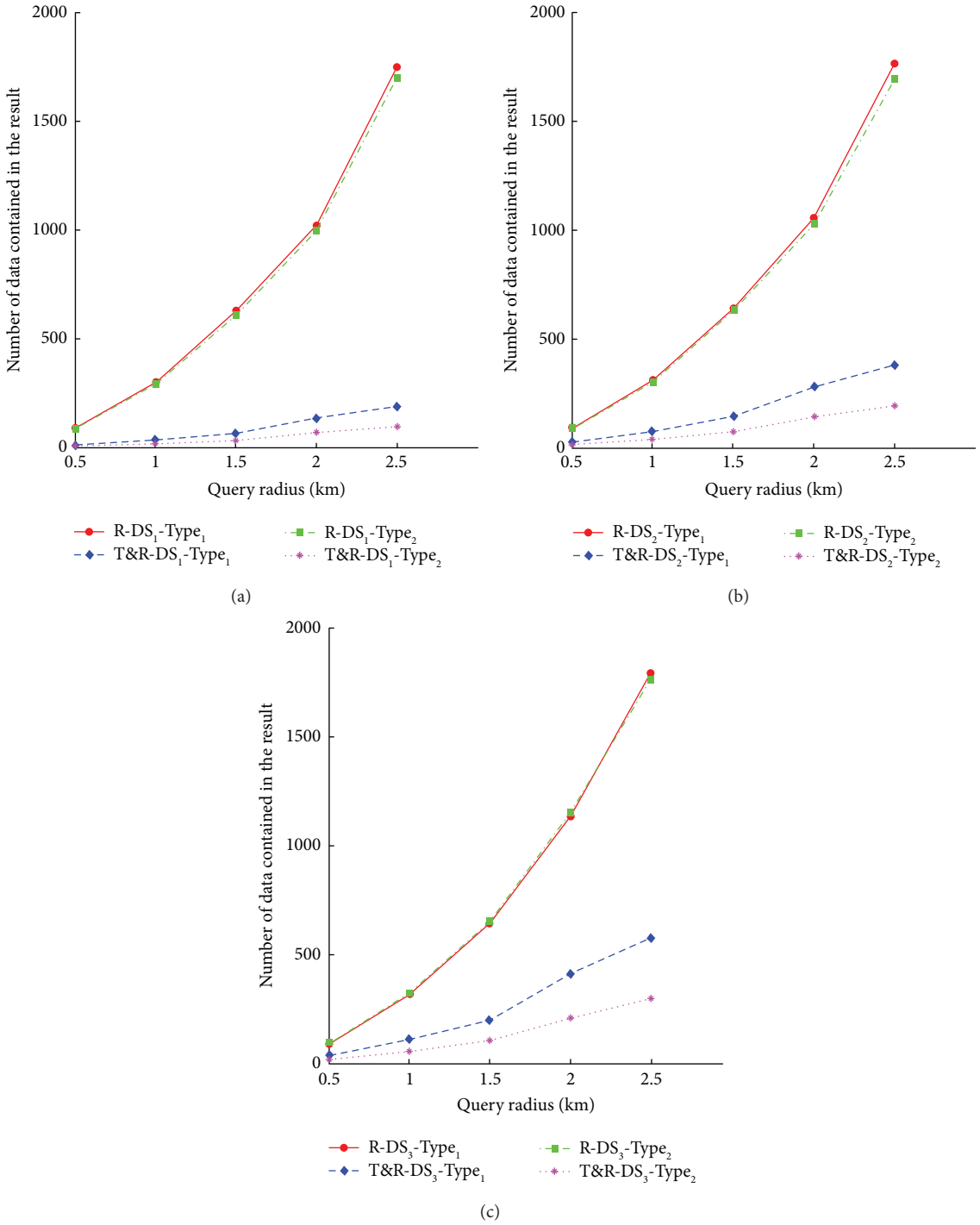

Figure 6: Number of data contained in the query result on different outsourced datasets. (a) Number of data contained in the query result on outsourced dataset. DS_1 (b) Number of data contained in the query result on outsourced dataset. DS_2(c) Number of data contained in the query result on outsourced dataset DS_3.

greater than that of the low-density POI type data in the query area. Accordingly, the cloud side needs to insert more data into the query result.

As mentioned above, DS_3 can be seen as a high-density outsourced dataset if DS_1 or DS_2 is assumed to be a low-density outsourced dataset. In the mode that supports both the type query and the range query (i.e., *T&R*), for any POI type (e.g., $Type_1$), it can be seen that the number of data contained in the query result on a high-density outsourced dataset (e.g., DS_3) is greater than that on a low-density outsourced dataset (e.g., DS_1). The reason is that a high-density outsourced dataset has a greater data density compared to a low-density dataset in the user query area, which leads to the increase of the data contained in the query result.

In the mode that only supports the range query (i.e., R), since the cloud side cannot find out the encrypted data according to the user's POI query type, the data inserted into the query result are the encrypted data within the user's query radius. Therefore, the number of data contained in the query result has similar behavior in each outsourced dataset (i.e., the number of data contained in the query result is almost the same in each outsourced dataset when the user's query radius is fixed). However, in the mode that supports both the type query and the range query (i.e., $T\&R$), the data inserted into the query result must meet two conditions: (1) the data are consistent with the user's POI query type and (2) the data are located in the user's query radius. Therefore, no matter on which outsourced dataset, the number of data contained in the query result under the mode that supports both the type query and the range query is less than that under the mode that only supports the range query. Therefore, since the number of data contained in the query result is related to the computation burden on the user side, the scheme (i.e., PPQS) that supports both the type query and the range query can effectively reduce the computation burden on the user side compared to the scheme that only supports the range query.

7. Conclusion

In this paper, an efficient privacy-preserving LBS query scheme (i.e., PPQS) in the outsourced cloud scenario is proposed. Specifically, we propose a dual cloud structure to enhance the ability of privacy preservation for sensitive information in the outsourced cloud, i.e., our scheme has the ability to resist the insider attack and the eavesdropping attack while preserving data privacy and query privacy. Moreover, by using the techniques of homomorphic encryption and searchable symmetric encryption, our scheme supports both the type query and the range query, which can effectively improve the data retrieval efficiency of the cloud side and reduce the computation burden on the user side and the cloud side. Finally, the effectiveness and performance of our scheme are validated through the analysis on security and computation cost and extensive simulations.

Data Availability

In this paper, the LBS data are collected from the public API interface of Amap service. The URL is https://lbs.amap.com/api/webservice/summary.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was supported in part by Yunnan Key Laboratory of Blockchain Application Technology (202105AG070005), Natural Science Foundation of China (61802005), Joint of Beijing Natural Science Foundation and Education Commission (KZ201810009011), Beijing Municipal Natural Science Foundation (M21029), Natural Science Foundation of China under Grant 61802005, and Talent Special Project (XN083).

References

[1] Y. Zou, Y. Chai, S. Shi et al., "Improved cloud-assisted privacy-preserving profile-matching scheme in mobile social networks," *Security and Communication Networks*, vol. 2020, no. 9, pp. 1–12, Article ID 4938736, 2020.

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

[3] L. Li, R. Lu, and C. Huang, "Eplq: efficient privacy-preserving location-based query over outsourced encrypted data," *IEEE Internet of Things Journal*, vol. 3, no. 2, pp. 206–218, 2015.

[4] H. Zhu, F. Liu, and H. Li, "Efficient and privacy-preserving polygons spatial query framework for location-based services," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 536–545, 2016.

[5] M. Zeng, K. Zhang, J. Chen, and H. Qian, "P3gq: a practical privacy-preserving generic location-based services query scheme," *Pervasive and Mobile Computing*, vol. 51, no. 1, pp. 56–72, 2018.

[6] S. Yang, S. Tang, and X. Zhang, "Privacy-preserving k nearest neighbor query with authentication on road networks," *Journal of Parallel and Distributed Computing*, vol. 134, no. 12, pp. 25–36, 2019.

[7] J.-M. Bohli, N. Gruschka, M. Jensen, L. L. Iacono, and N. Marnau, "Security and privacy-enhancing multicloud architectures," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 4, pp. 212–224, 2013.

[8] F. Fan, S. Li, H. Dai, C. Hu, W. Dou, and Q. Ni, "A k-anonymity based schema for location privacy preservation," *IEEE Transactions on Sustainable Computing*, vol. 13, no. 8, pp. 16–29, 2017.

[9] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, pp. 13–22, 2007.

[10] B. Niu, Z. Zhang, X. Li, and H. li, "Privacy-area aware dummy generation algorithms for location-based services," in *Proceedings of the 2014 IEEE International Conference on Communications, ICC*, pp. 957–962, IEEE, ydney, NSW, Australia, 10 June 2014.

[11] H. Lian, W. Qiu, Di Yan, J. Guo, Z. Li, and P. Tang, "Privacy-preserving spatial query protocol based on the moore curve for location-based service," *Computers & Security*, vol. 96, no. 3, pp. 101–125, 2020.

[12] M. D. Firoozjaei, J. Yu, H. Choi, and H. Kim, "Privacy-preserving nearest neighbor queries using geographical features of cellular networks," *Computer Communications*, vol. 98, no. 5, pp. 11–19, 2016.

[13] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of the IEEE 36th Annual Foundations of Computer Science*, pp. 41–50, IEEE, NW Washington, DC. United States, 23 October 1995.

[14] P. Russell, Md G. Kaosar, X. Yi, and E. Bertino, "Privacy-preserving and content-protecting location based queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1200–1210, 2013.

[15] X. Yi, P. Russell, E. Bertino, and V. Varadharajan, "Practical k nearest neighbor queries with location privacy," in *Proceedings of the 2014 IEEE 30th International Conference on Data*

Engineering, pp. 640–651, IEEE, Chicago, IL, USA, 31 March 2014.

[16] J. Shao, R. Lu, and X. Lin, "FINE: a fine-grained privacy-preserving location-based service framework for mobile devices," in *Proceedings of the IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pp. 244–252, IEEE, Toronto, ON, Canada, 27 April 2014.

[17] O. Lu, H. Yin, Q. Zheng, S. Xiao, G. Yang, and Y. Hu, "An efficient and privacy-preserving multiuser cloud-based LBS query scheme," *Security and Communication Networks*, vol. 2018, pp. 1–11, Article ID 4724815, 2018.

[18] P. Pascal, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the International conference on the theory and applications of cryptographic techniques*, pp. 223–238, Springer, Prague, Czech Republic, 2 May 1999.

[19] P. Golle, "A private stable matching algorithm," in *Proceedings of the 10th International Conference of Financial Cryptography and Data Security*, pp. 65–80, Springer-Verlag, Anguilla British West Indies, 27 February 2006.

[20] C. Xu, X. Xie, L. Zhu, K. Sharif, and M. Guizani, "PPLS: a privacy-preserving location-sharing scheme in mobile online social networks," *Science China Information Sciences*, vol. 63, no. 3, pp. 105–122, 2020.

[21] H. Lipmaa, "Verifiable homomorphic oblivious transfer and private equality test," in *Proceedings of the 9th International Conference on the Theory and Application of Cryptology CiteSeer*, pp. 25–40, dblp, Taipei, Taiwan, 30 November 2003.

[22] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. . Roşu, and M. Steiner, *Highly-scalable Searchable Symmetric Encryption with Support for Boolean Queries*, Springer, Berlin/Heidelberg, Germany, 2013.

[23] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[24] K. Xue, S. Li, J. Hong, Y. Xue, N. Yu, and P. Hong, "Two-cloud secure database for numeric-related SQL range queries with privacy preserving," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1596–1608, 2017.

[25] "lbs.amap.com. Api of webservice," 2019, https://lbs.amap.com/api/webservice/summary.