Cyber Security and Applications 1 (2023) 100007

Contents lists available at ScienceDirect

Cyber Security and Applications

journal homepage: http://www.keaipublishing.com/en/journals/cyber-security-and-applications/

PPT-LBS: Privacy-preserving top-k query scheme for outsourced data of location-based services

Yousheng Zhou[a], Xia Li[a,*], Ming Wang[b], Yuanni Liu[a]

[a] School of Cyber Security and Information Law, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[b] College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

ARTICLE INFO

Keywords:
Privacy protection
Location-based services
Top-k query
Outsourcing computing

ABSTRACT

Location-based service (LBS) is enjoying a great popularity with the fast growth of mobile Internet. As the volume of data increases dramatically, an increasing number of location service providers (LSPs) are moving LBS data to cloud platforms for benefit of affordability and stability. However, while cloud server provides convenience and stability, it also leads to data security and user privacy leakage. Aiming at the problems of insufficient privacy protection and inefficient query in the existing LBS data outsourcing schemes, this paper presents a novel privacy-preserving top-k query for outsourcing situations. Firstly, to ensure data security of LSP and privacy of the user, the enhanced asymmetric scalar-product preserving encryption and public key searchable encryption have been adopted to encrypt outsourced data and LBS query, which can effectively lower the computational cost and realize the privacy protection search. Secondly, an efficient and secure index structure is constructed by using a coded quadtree and the bloom filter, so that the cloud server can quickly locate the user's query region to improve retrieval efficiency. Finally, the formal security analysis is given under the random oracle model, and the performance is evaluated by experiments which demonstrates that our scheme is preferable to existing schemes.

1. Introduction

Location-based service (LBS) has been used in many fields, such as military, medical treatment, emergency rescue, etc., due to the rapid popularity of mobile devices [1]. However, as the upsurge of LBS dataset, LBS data's high storage and computation costs produce a heavy burden on location service providers (LSPs). The rapid development of cloud computing provides a new operation mode for LBS, that is, LSP uploads a large amount of LBS data onto the cloud to process user's queries with its powerful computing power, which effectively reduces LSP cost. However, cloud computing brings data security and user privacy problems while facilitating data computing and storage. In a traditional computing mode, the user's data is usually on the LSP-controlled or trusted platform for processing. Still, after outsourcing data to a cloud server (CS), the data's physical control capability is handed over to the cloud. In the outsourcing environment, the CS is usually supposed to be "semi-honest", it will perform the user's query request honestly. Meanwhile, it also attempts to derive useful information from the user's query and the stored data [2–5]. Hence, LBS data secure storage and computation in an untrusted cloud environment have become a critical issue that needs to be resolved urgently.

To achieve the LBS system's privacy protection in an outsourcing environment, researchers present a series of location privacy protection methods. Zeng et al. [6] presented a new search scheme based on enhanced asymmetric scalar-product preserving encryption algorithm and encrypted inverted index technology to support generic LBS query over encrypted data for cloud environment. In this scheme, the user can specify the geographical scope and search keywords. After searching, the CS returns the point of interest (POI) records which matches the given area and keywords. Yang et al. [7] presented a verifiable privacy protection scheme for kNN query under road network environment based on Voronoi diagram, 2-Hop tag index, and some cryptographic primitive, which could simultaneously preserve the privacy of spatial data and kNN query, and verify the reliability of query results. Xie and Wang [8] put forward an LBS privacy protection scheme based on attribute encryption of ciphertext policy (CP-ABE), which calculate and compare location distance for the authorized user without revealing its privacy. Zhu et al. [9,10], aiming at the privacy and efficiency problems in the LBS range query service process, designed an LBS circle region and polygon region range query scheme with efficient privacy protection by using an improved homomorphic encryption algorithm, which can provide query service while ensuring the user's query privacy and the data confidentiality of LSP. However, the existing research on privacy protection for LBS data outsourcing mainly focuses on the geographic scope or interest keywords query and rarely supports top-k query. Besides, the retrieval algorithm of LBS data fails to protect the user's search mode.

* Corresponding author.
E-mail address: s190802004@stu.cqupt.edu.cn (X. Li).

https://doi.org/10.1016/j.csa.2022.100007
Received 19 April 2022; Received in revised form 10 June 2022; Accepted 18 August 2022
Available online 23 August 2022
2772-9184/© 2022 The Authors. Published by Elsevier B.V. on behalf of KeAi Communications Co., Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

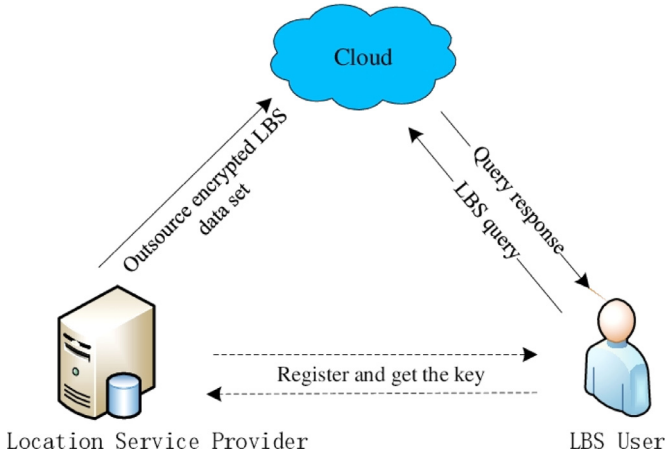


Fig. 1. The system model.

Our contributions are outlined as follow:

1) Firstly, we present a top-k query scheme for LBS outsourced data. To protect the LBS data and user's query against the attacker, the enhanced asymmetric scalar-product preserving encryption and public key searchable encryption are utilized to construct our secure query scheme while supporting accurate top-k search over the encrypted LBS data.

2) Secondly, we construct a secure index structure based on coded quadtree and bloom filter, which enables the CS to fast locate the user's query region with the user's query request, thereby improving the retrieval efficiency.

3) Lastly, we conduct formal security proof and experiments based performance analysis, which exhibit that our presented scheme is secure and efficient.

Organization. The remainder of our thesis is organized as follows. Section 2 introduces the system model and design goals. In Section 3, we give some preliminaries. Then, the detailed construction of the proposed scheme is illustrated in Section 4. The security analysis and performance evaluation are demonstrated in Section 5 and Section 6, respectively. Finally, the paper is concluded in Section 7.

2. Problem formulations

2.1. System model

Our scheme aims to provide the user with secure and efficient query services while ensuring LBS data security and user query privacy. There are three entities: the location service provider (LSP), the cloud server(CS), and the LBS user, as shown in Fig. 1.

- **Location Service Provider (LSP):** An LSP owns a large number of LBS resources. It outsources massive LBS data to the CS to benefit from cheap storage and reliable computation services. To guarantee LBS data confidentiality, each LBS data will be encrypted at first and then uploaded to the cloud. Besides, LSP also provides registration service for the LBS user. Once the user passes registration, LSP sends an authentication certificate and the key to the user via a secure communication channel.
- **Cloud Server (CS):** CS has abundant storage and computing resources, it is responsible for storing ciphertext data sets from LSP and providing LBS query services for users.
- **LBS User:** A LBS user first registers with LSP to obtain the key. In order to protect privacy, the query requests of users are submitted to the CS in the way of trapdoor.

2.2. Design goals

In this paper, it is assumed that CS is "honest but curious". In other words, it will execute the user's query request honestly, but it also tries to study useful information from the user's query and stored LBS data. As with other related studies [2,3,6,7,9], LSP and user are assumed to be honest, and LSP and user do not conspire with CS to obtain other user's privacy. Therefore, to protect the LBS data and the user's query, our scheme achieves the following security objectives:

(1) Confidentiality: The CS cannot understand any content of LBS data stored by LSP. The outsourced LBS data is encrypted to prevent the CS from obtaining any valid information from the data set.

(2) Privacy: The user's query should be confidential to the CS since it contains private information, such as area and personal interest. To ensure that the user's query privacy is not disclosed, this paper encrypts the user's query request and submits it to CS in the form of a query trapdoor, thus preventing CS from obtaining any useful information about the user's query.

3. Preliminaries

Some basics used in our scheme are introduced in this section, including bilinear pairing map, hard problem assumptions, and bloom filter.

3.1. Bilinear pairing map

$\mathbb{G}$ and $\mathbb{G}_T$ are two q-order multiplication cyclic groups, with the generator g of $\mathbb{G}$. $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map, and it has the follow properties:

(1) Bilinearity: for any $x, y \in \mathbb{Z}_q^*$, there is $e(g^x, g^y) = e(g, g)^{xy}$.

(2) Non-degenerate: $e(g, g) \neq 1$.

(3) Computability: there exists a probabilistic polynomial-time (PPT) algorithm to evaluate $e(g, g)$.

3.2. Hard problem assumptions

Decisional Bilinear Diffie-Hellman (DBDH) Assumption: Assume that there is a q-order group $\mathbb{G}$ and a generator g. Provided with $g^a, g^b, g^c \in \mathbb{G}, Z \in \mathbb{G}_T$, where $a, b, c \in \mathbb{Z}_q^*$, the DBDH problem is to decide $Z \stackrel{?}{=} e(g, g)^{abc}$.

m Decisional Linear (mDLIN) Problem: The mDLIN problem is a variant of the DLIN problem. Assume that there is a group $\mathbb{G}$, where the order is q and the generator is g. Provided with $g^a, g^b, g^{ra}, g^{t/b}, g^c$, where a, b, r, t, c are randomly chosen from $\mathbb{Z}_q^*$. mDLIN problem is to determine $g^c \stackrel{?}{=} g^{r+t}$.

3.3. Bloom filter

The bloom filter can easily determine whether an element belongs to the current set, the core is a bit array and t independent hash function $H_i : \{0,1\}^* \to \{1, 2, \ldots, \ldots, m\}, i \in [1, t]$, the initial state of the values in the array is 0. Given a set $X = \{x_1, x_2 \cdots x_n\}$, select t hash functions to map each element to the bloom filter and set the bit position of each generated hash value to 1.

When verifying whether an certain element $x_i, i \in [1, n]$ exists in the set X, use t hash function to map x_i to the bloom filter. If one of the corresponding positions is 0, then x_i must not belong to X.If each of these corresponding positions is 1, then maybe x_i is a member of X, as shown in Fig. 2.

The error rate p of the bloom filter is defined as:

$$p = 1 - (e^{\frac{-tn}{m}})^t \tag{1}$$

where m is the length of bit array, n denotes the size of X,and t denotes the amount of hash functions [3].

4. Construction of PPT-LBS

The concrete construction of our proposed scheme is described in this section. Our scheme is improved based on Zhao et al. [11], which

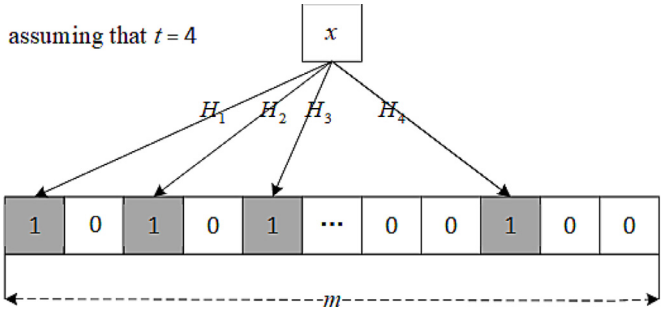


Fig. 2. Schematic diagram of bloom filter.

Table 1
Symbols definition.

Notations	Definition
$\boldsymbol{M}_1, \boldsymbol{M}_2$	invertible matrix
l	matrix dimension, set $l = 80$ in this scheme
$b \in \{0,1\}^l$	a bit string of length l
$((x, y), w)$	one POI data
$((x_u, y_u), R)$	the user's coordinate is (x_u, y_u) and the query radius is R
w_q	user's query keyword
k	the user want to query the number of POIs closest to his/her location
Q	the intersection region
$\mathfrak{R}^*$	a set of all expanded code sequences

enables users can perform keyword query of POI and top-k query. For example, Alice wants to query the three restaurants closest to him/her within 500 m nearby. At the same time, under the premise of ensuring that her location and query keywords are not leaked, the CS returns the POI records that meet Alice's conditions.

The proposed scheme is made up of eight phases: system initialization, key generation, user registration, LBS data encryption, index construction, LBS query generation, LBS data retrieval, respectively. Symbols used are defined as in Table 1.

4.1. System setup

System setup is used to generate some public parameters, which are performed by LSP. With the following steps:

(1) LSP selects two multiplication cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ with the large prime order q. Let g be a generator of $\mathbb{G}$ and a bilinear pairing map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$;

(2) LSP selects a one-way hash function $H_0 : \{0,1\}^* \to G$ and a hash family with t hash functions $H_i : \{0,1\}^* \to \{1, 2, \ldots, \ldots, m\}, i \in [1, t]$, where m denotes the size of the bloom filter, whose main function is to map a bit string into the bloom filter vector.

LSP open public parameters $param = \{\mathbb{G}, \mathbb{G}_T, g, H_0, H_i\}, i \in [1, t]$.

4.2. Key generation

(1) LSP selects two $l \times l$ dimensions invertible matrix $\boldsymbol{M}_1, \boldsymbol{M}_2$, a bit string b of length l and $l-3$ random number $\{\alpha_j\}_{j\in[1,l-3]}$. Here the matrix dimension l should be big enough to resist brute force attacks, matrix dimensions are set $l = 80$;

(2) $key = \{\boldsymbol{M}_1, \boldsymbol{M}_2, b, \{\alpha_j\}_{j\in[1,l-3]}\}$ is stored as a secure key in LSP, which is used for user to generate trapdoor and decrypt data ;

(3) LSP randomly chooses $x_L \in \mathbb{Z}_q^*$ and sets its private key $SK_L = x_L$, the public key is $PK_L = g^{x_L}$.

4.3. User registration

When a new user join the LBS system, LSP registers the user at this phase.

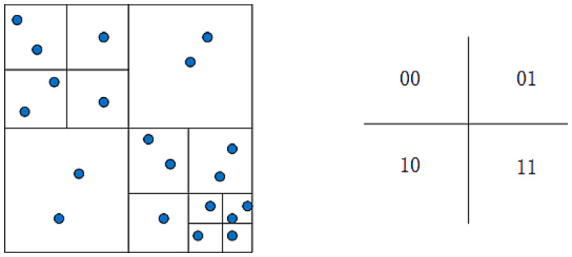


Fig. 3. Map devision.

(1) The user randomly selects $x_u \in \mathbb{Z}_q^*$ and sets its private key $SK_u = x_u$, the public key is $PK_u = g^{x_u}$;

(2) The user calculates registration information $MID_u = H_0(ID_u)$, and send it to the LSP for registration request;

(3) When the LSP receives the registration request message sent by the user, LSP computes an authentication certificate $C_u = e((MID_u)^{SK_L}, g)$ for the user and send the key to the user via secure communication channel;

(4) LSP sends the C_u to the CS, which stores it in the user list for subsequent user authentication.

4.4. LBS Data encryption

The POI data set stored in the LSP is represented as $\boldsymbol{DB} = \{d_1, d_2, \ldots, d_n\}$. To be simple, use $d_i = \{(x, y), w\}$ to represent one of POI record, where (x, y) means the coordinates of the POI, w shows the keyword of the POI(e.g., restaurants, hotels, bars, etc.). In order to ensure data confidentiality, LSP outsources data set $\boldsymbol{DB}$ to the CS in the form of ciphertext.

(1) Coordinate Encryption

For POI coordinate (x, y), LSP generates an l-dimensional vector $\hat{\boldsymbol{p}}$, where the first three dimensions of data is $(x, y, -0.5(x^2 + y^2))$. For $j \in [4, l]$, set $\hat{\boldsymbol{p}}[j] = \alpha_{j-3}$. Then, $\hat{\boldsymbol{p}}$ is split according to the bit string b into two l dimensional vector $\hat{\boldsymbol{p}}_a$ and $\hat{\boldsymbol{p}}_b$:

- if $b[i] = 1, i \in [1, l]$, set $\hat{\boldsymbol{p}}_a[i] + \hat{\boldsymbol{p}}_b[i] = \hat{\boldsymbol{p}}[i]$;
- if $b[i] = 0, i \in [1, l]$, set $\hat{\boldsymbol{p}}_a[i] = \hat{\boldsymbol{p}}_b[i] = \hat{\boldsymbol{p}}[i]$.

Calculate:$\boldsymbol{C}_a = \boldsymbol{M}_1{}^T \hat{\boldsymbol{p}}_a, \boldsymbol{C}_b = \boldsymbol{M}_2{}^T \hat{\boldsymbol{p}}_b$.

Output coordinat ciphertext: $C_{cord} = \{\boldsymbol{C}_a, \boldsymbol{C}_b\}$.

(2) Keyword Encryption

LSP randomly selects $r \in \mathbb{Z}_q^*$, and derives the keywords ciphertext $C_w = \{C_{w_1}, C_{w_2}\}$, where

$$C_{w_1} = H_0(w)^{SK_L} \cdot g^r, \tag{2}$$

$$C_{w_2} = PK_u{}^r. \tag{3}$$

So far, a piece of POI ciphertext data C_d is represented as: $C_d = \{C_{cord}, C_w\}$.

LSP encrypts all POI data in $\boldsymbol{DB}$, and the ciphertext data set is represented as: $\boldsymbol{EDB} = \{C_{d_i}\}, i \in [1, n]$.

The LSP uploads the ciphertext data set $\boldsymbol{EDB}$ to the CS.

4.5. Index construction

(1) Binary coding of geographic data

LSP recursively divides the map into four regions until the number of POI stored in each subregion does not exceed a specified threshold. At the same time, LSP selects 00,01,10,11 to denote four regions, each of which can be represented by a bit string, as shown in Fig. 3.

For example, in Fig. 4, the entire region is divided into four subregions, encode as 00,01,10,11. The upper right subregion with the bit string "01" is further divided into four subregions encoded as 0100, 0101, 0110, 0111, and the bit string "0101" region is divided again, and so on so that each region is a unique bit string encode.

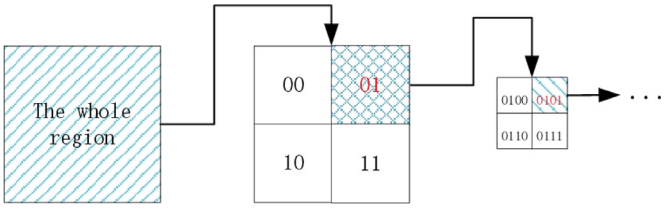


Fig. 4. Hierarchical coding of a region.

(2) Coded quadtree index construction

The LSP uses a quadtree to store bit string and POI ciphertext data of the above-divided regions. In this coded quadtree index structure, the root node denotes the entire region, and each non-leaf node has 4 child nodes denoting four subregions. Non-leaf nodes store bit strings of each region, and leaf nodes store the ciphertext of POI within the region. For a given POI data d, LSP first encrypts it to C_d, then find its subregion encode, and finally adds it to the corresponding leaf node of the subregion of the coded quadtree.

Take the example of inserting data in the region of the bit string as "001001". It starts from the root node to search the data record pertaining to the subregions 00, 010, 01001, respectively. Afterwards, the data record is interjected into subregion 001001, as shown in Fig. 5.

4.6. LBS Query generation

An LBS user's query request can be expressed as: $\{((x_u, y_u), R), w_q, k\}$, where $((x_u, y_u), R)$ is the user's current coordinates and the query radius, w_q is the user's query keyword, k is user want to query the number of POIs closest to his/her location. In the cloud-based LBS system, in order to prevent the private query being disclosed, the user will send the query in the form of trapdoor.

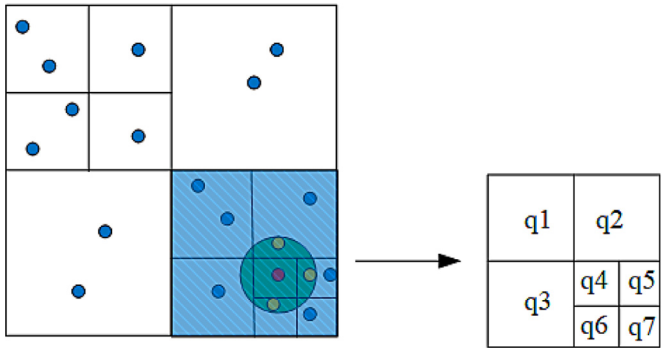


Fig. 6. Example of user's query.

(1) Index trapdoor V_{BF} generation

First of all, the user computes the circular region as the query range according to the current position (x_u, y_u) and the query radius R, as shown in the shaded green part of Fig. 6. The intersecting region of the user's query range and the subregions divided is expressed as $Q = \{q_1, q_2, \ldots, q_k\}$, as shown in the blue shaded part of Fig. 6.

Once Q is determined, i.e.,$Q = \{q_1, q_2, \ldots, q_7\}$, it can be expanded into a series of bit strings by listing all prefix substrings of $q_i, i \in [1, 7]$ for every two neighbouring bits.

For instance, suppose a bit string is 111101, the extended code sequence is represented as $\mathfrak{R}_i$, $\mathfrak{R}_i = \{11, 1111, 111101\}$. User combine the extended code sequence of all query subregions $q_i (i \in [1, k])$ to get a set of code sequence, $\mathfrak{R}^* = \bigcup_{i=1}^{i=k} \mathfrak{R}_i$. For every bit string in $\mathfrak{R}^*$, the user uses t hash functions $H_i : \{0, 1\}^* \rightarrow \{1, 2 \ldots \ldots m\}$, $i \in [1, t]$ to hash each bit string sequence in $\mathfrak{R}^*$ to get an index trapdoor $\boldsymbol{V}_{BF}$.

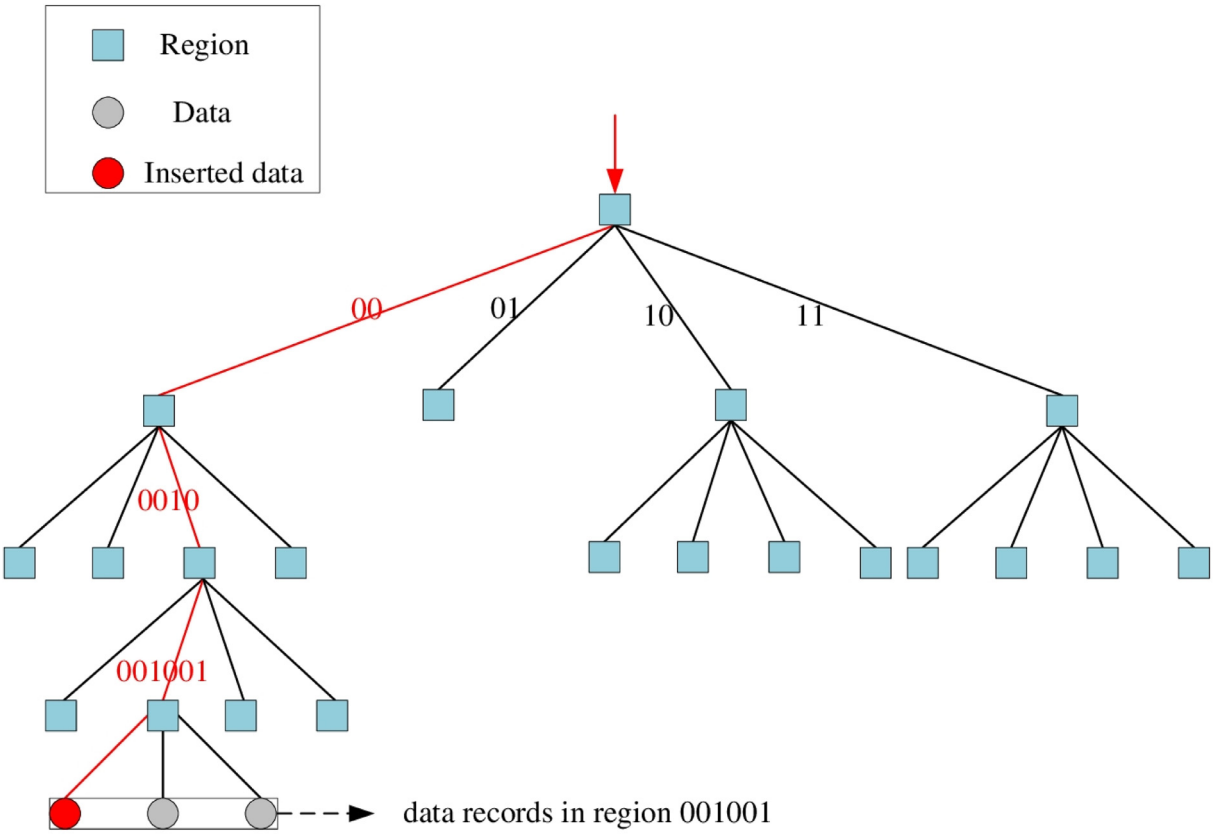


Fig. 5. Example of data insertion.

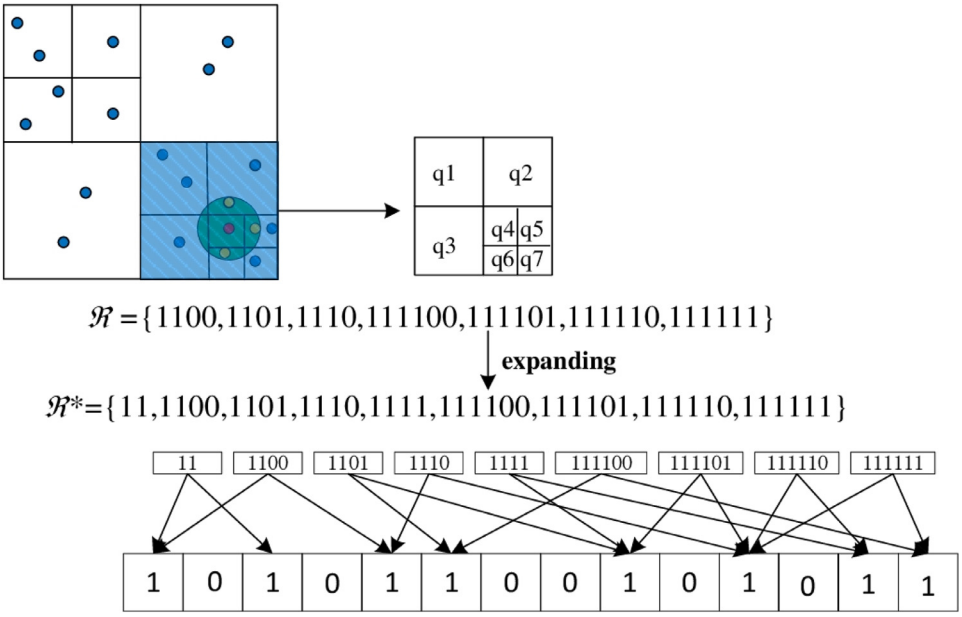


Fig. 7. Example of range derivation.

Take an example in Fig. 7. Assume that the current coordinate of the user is (26, 81). The user first determines the query range according to the query radius R, which is represented as the green circular region, and its intersection with the subregion divided is the blue shaded part, which is represented as $Q = \{q_1, \ldots\ldots, q_5\}$. The bit string of the seven subregions is $\mathfrak{R} = \{1100, 1101, 1110, 111100, 111101, 111110, 111111\}$. The user then expands these bit strings into collections $\mathfrak{R}^* = \{11, 1100, 1101, 1110, 1111, 111100, 111101, 111110, 111111\}$. Finally, the user uses t hash functions to hash all the bit strings in $\mathfrak{R}^*$ to obtain index trapdoor $\boldsymbol{V}_{BF} = 10101100101011$. **(2) Generation of coordinate trapdoor**

User randomly selects a positive integer λ, computes the first 3 dimensions of data $(\lambda x_u, \lambda y_u, \lambda)$. For $j \in [4, l\text{-}1]$, set $\hat{\boldsymbol{q}}[j] = \varpi_j$($\varpi_j$is a random number). For the last dimension, set $\hat{\boldsymbol{q}}[l] = \frac{-\sum_{i=4}^{l-1} \alpha_{i-3} \varpi_i}{\alpha_{i-3}}$. Then, $\hat{\boldsymbol{q}}$ is split according to the bit string b into two l dimensional data $\hat{\boldsymbol{q}}_a$ and $\hat{\boldsymbol{q}}_b$:

- if $b[i] = 0, i \in [1, l]$, set $\hat{\boldsymbol{q}}_a[i] + \hat{\boldsymbol{q}}_b[i] = \hat{\boldsymbol{q}}[i]$;
- if $b[i] = 1, i \in [1, l]$, set $\hat{\boldsymbol{q}}_a[i] = \hat{\boldsymbol{q}}_b[i] = \hat{\boldsymbol{q}}[i]$.

Calculate:$\boldsymbol{T}_a = \boldsymbol{M}_1{}^{-1}\hat{\boldsymbol{q}}_a, \boldsymbol{T}_b = \boldsymbol{M}_2{}^{-1}\hat{\boldsymbol{q}}_b$.

Output coordinat trapdoor: $T_L = \{\boldsymbol{T}_a, \boldsymbol{T}_b\}$.

(3) Generation of keyword trapdoor

Since the keyword searched by the user may expose sensitive information such as interests, hobbies, behavior habits, etc., the user needs to encrypt the search keyword before sending the query request. User first chooses their query keyword w_q, and then randomly choose $r' \in \mathbb{Z}_q^*$ to encrypt w_q into $T_w = \{T_{w_1}, T_{w_2}, T_{w_3}\}$, where

$$T_{w_1} = e(H_0(w_q)^{SK_u}, PK_L{}^{r'}), \tag{4}$$

$$T_{w_2} = g^{r'}, \tag{5}$$

$$T_{w_3} = PK_u{}^{r'}. \tag{6}$$

The user sends the query trapdoor $\boldsymbol{TRAPDOOR} = \{MID_u, \boldsymbol{V}_{BF}, T_L, T_w, k\}$ to the CS.

4.7. LBS Data retrieval

Once the CS receives ***TRAPDOOR*** from the user, it first verifies the user's identity and determines whether the $C_u = e(PK_L, MID_u)$ is established. If not, CS rejects the query; otherwise, CS searches user query. Specific steps are as follows:

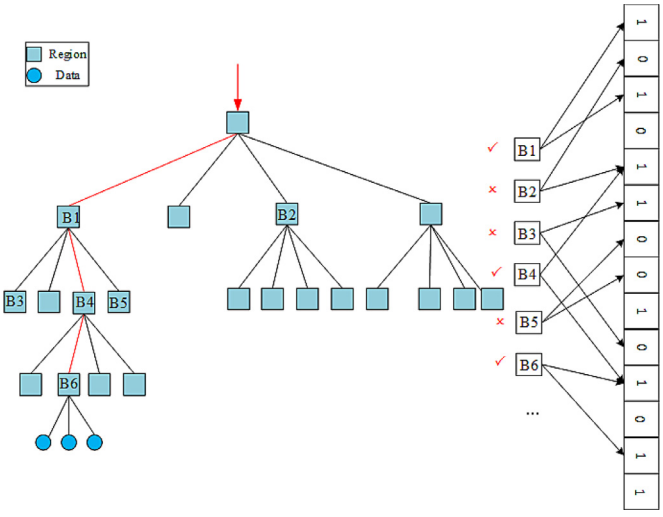


Fig. 8. Example of coordinate matching.

(1) Coordinate matching

After receiving the $\boldsymbol{V}_{BF}$ in the query trapdoor from the user, CS utilizes it to search the coded quadtree. In the coded quadtree, each non-leaf node is denoted by a bit string. CS examines if the mapping positions of B_i of the node are all "1" in $\boldsymbol{V}_{BF}$. If it is, CS subsequently search down the encoding quadtree; otherwise, CS changes the branch to carry on the search until all data in $\mathfrak{R}^*$ is hit, as shown in Fig. 8. CS stores these data in the temporary resource list (TRL) and then goes to step (2); **(2) Keyword matching**

Verify whether data $T_{w_1} \cdot e(C_{w2}, T_{w_2}) = e(C_{w_1}, T_{w_3})$ in the TRL is established. If it is established, it means that the POI record meets the user's query keyword, and go to step (3); otherwise, select the next POI record for matching;

(3) top kPOI closest

Suppose $\{C_{1a}, C_{1b}\}$ and $\{C_{2a}, C_{2b}\}$ are the ciphertext data of the data POI_1 and POI_2 respectively, $\{T_a, T_b\}$ is the trapdoor of the user query. Determine whether $(\boldsymbol{C}_{1a} - \boldsymbol{C}_{2a})\boldsymbol{T}_a + (\boldsymbol{C}_{1b} - \boldsymbol{C}_{2b})\boldsymbol{T}_b > 0$ is true. If it is true, the POI_1 is closer to the user than the POI_2.

Finally, CS performs distance comparison and sorting, and returns the first k encrypted POI to the user.

4.8. User decryption

User get k specified POI ciphertext records from the CS, expressed as $\{\boldsymbol{C}_{ai}, \boldsymbol{C}_{bi}\}, i \in [1,k]$. For one of the POI ciphertext records, the user first calculates:

$$\hat{\boldsymbol{p}}_a = \pi_d(\boldsymbol{M}_1^T)^{-1} \times \boldsymbol{C}_a, \tag{7}$$

$$\hat{\boldsymbol{p}}_b = \pi_d(\boldsymbol{M}_2^T)^{-1} \times \boldsymbol{C}_b, \tag{8}$$

where $\pi_d = (I_d, 0)$ is a 2×80 matrix, I_d is a 2×2 identity matrix.

For $\hat{\boldsymbol{p}}_a$ and $\hat{\boldsymbol{p}}_b$, if $b_t = 1, t \in [1,2]$, set $\hat{\boldsymbol{p}}_o[t] = \hat{\boldsymbol{p}}_a[t] + \hat{\boldsymbol{p}}_b[t]$; else set $\hat{\boldsymbol{p}}_o[t] = \hat{\boldsymbol{p}}_a[t] = \hat{\boldsymbol{p}}_b[t]$. The real coordinates of POI is $(x = \hat{\boldsymbol{p}}_o[1], y = \hat{\boldsymbol{p}}_o[2])$.

5. Security analysis

5.1. Data confidentiality

For each LBS data, LSP outsources it to the CS in the form of ciphertext. Specifically, for the coordinates of POI record, we use the enhanced ASPE algorithm [12] to encrypt it. In the known ciphertext model, if CS wants to get the real value of POI coordinates, it must restore two matrix $\boldsymbol{M}_1, \boldsymbol{M}_2$ from the ciphertext and correctly guess the bit string b. For CS, the equations used to determine the transformation matrices are:$\boldsymbol{C}_a = \boldsymbol{M}_1^T \times \hat{\boldsymbol{p}}_a$ and $\boldsymbol{C}_b = \boldsymbol{M}_2^T \times \hat{\boldsymbol{p}}_b$, where $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are two unknown $l \times l$ dimensional matrixs, there are 2^{l^2} unknowns in $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$. The vector $\hat{\boldsymbol{p}}_a, \hat{\boldsymbol{p}}_b$ are split from the l-dimensional bit string b, which has $2l$ unknowns. Since given only $2l$ equations, which are less than the amount of unknowns, the transformation matrix cannot be solved by the adversary without enough information. Also, Wong et al. [12] security analysis shows that when l=80, its security is equivalent to that of the 1024-bit key RSA encryption algorithm. Thus, under the premise of key security, the CS cannot restore the real location of the POI through ciphertext data.

Lemma 1. *Assuming that the problem of mDLIN is hard, then for any probability polynomial time adversary $\mathcal{A}$, the probability advantage $\mathrm{Adv}_{\mathcal{A}}^{C}(\lambda)$ of distinguishing the keyword ciphertext is negligible.*

Proof 1. Assuming that exists an adversary $\mathcal{A}$ that can correctly distinguish the keyword ciphertext with a non-negligible probability advantage ϵ_C. We demonstrate that the challenger C can solve the mDLIN problem with a non-negligible advantage ϵ_C. Given an instance of the mDLIN problem with parameters $(g, g^a, g^b, g^{ra}, g^{t/b}, W) \in \mathbb{G}$, where $a, b, r, t \in \mathbb{Z}_q^*$, the goal of C is to determine $W = g^{r+t}$ or a random element of $\mathbb{G}$. C set $\mu \in \{0,1\}$, if $W = g^{r+t}$, μ=0; if W is random, μ=1.

(1) Initialization:

The Challenger C sets $(PK_u, PK_L) = (g^a, g^b)$ and sents it to the adversary $\mathcal{A}$ along with the public parameter $param = (\mathbb{G}, \mathbb{G}_T, e, q, g)$.

(2) Query:

$\mathcal{A}$ can ask C for the following query:

a) Hash query $\mathcal{O}_H$: $\mathcal{A}$ sends the keyword w_i to C for hash query. C will do the following:

① C maintains an initially empty list $L_H\langle w_i, h_i, \alpha_i, \tau_i\rangle$ for $\mathcal{A}$'s query, and if w_i already appears in L_H, C sends a response $H(w_i) = h_i$ to $\mathcal{A}$.

② Otherwise, C produces a random number $\tau_i \in \{0,1\}$ in probability such that $Pr[\tau_i = 0] = \theta$.

If $\tau_i = 0$, randomly chooses $\alpha_i \in \mathbb{Z}_q^*$ and computes $h_i = g^{t/b} \cdot g^{\alpha_i}$; Otherwise, sets $h_i = g^{\alpha_i}$.

③ C stores the tuple $\langle w_i, h_i, \alpha_i, \tau_i\rangle$ in L_H and returns $H(w_i) = h_i$ to $\mathcal{A}$.

b) Ciphertext query $\mathcal{O}_C$: $\mathcal{A}$ sends the keyword w_i to C for ciphertext query and C retrieves $\langle w_i, h_i, \alpha_i, \tau_i\rangle$ from the list L_H. If $\tau_i = 0$, it randomly outputs a bit $\mu' \in \{0,1\}$ as its guess of μ. Otherwise, it randomly selects $k_i \in \mathbb{Z}_q^*$ and computes the ciphertext $C_w = (C_1, C_2) = ((g^{\alpha_i})^b \cdot g^{k_i} = g^{\alpha_i b + k_i}, (g^a)^{k_i})$. C then returns C_w to $\mathcal{A}$.

c) Trapdoor query $\mathcal{O}_T$: $\mathcal{A}$ sends the keyword w_i to C for trapdoor query and C retrieves $\langle w_i, h_i, \alpha_i, \tau_i\rangle$ from the list L_H. If $\tau_i = 0$, it randomly outputs a bit $\mu' \in \{0,1\}$ as its guess of μ. Otherwise, it randomly selects $r_i \in \mathbb{Z}_q^*$ and derives the trapdoor $T_w = \{T_1, T_2, T_3\} = \{e(g^{\alpha_i a}, g^{br_i}) = e(H(w_i)^a, g^{br_i}), g^{r_i}, g^{ar_i}\}$. C then returns T_w to $\mathcal{A}$.

(3) Challenge:

After polynomial queries, $\mathcal{A}$ sends two keywords w_0^*, w_1^* to C, which have not been queried to $\mathcal{O}_T$ nor $\mathcal{O}_C$. Then, C performs the following operations:

① C performs a Hash query on w_0^*, w_1^* respectively: $H(w_0^*) = h_0^*$ and $H(w_1^*) = h_1^*$. w_0^*, w_1^* correspond to tuples $\langle w_0^*, h_0^*, \alpha_0^*, \tau_0^*\rangle$ and $\langle w_1^*, h_1^*, \alpha_1^*, \tau_1^*\rangle$, respectively. If τ_0^*, τ_1^* are both 1, C aborts the query and randomly outputs $\mu' \in \{0,1\}$ as its guess of μ;

② If at least one of τ_0^* and τ_1^* is 0, set $\hat{\mu}$ be the bit such that $\tau_{\hat{\mu}}^* = 0$. C computes the ciphertext $C^* = (C_1^*, C_2^*) = (W \cdot (g^{\alpha_{\hat{\mu}}^*})^b \cdot g^{k_i}, g^{ak_i})$. If $W = g^{r+t}$, then $C_1^* = g^{r+t} \cdot g^{b\alpha_{\hat{\mu}}^*} \cdot g^{k_i} = g^{(r+k_i)+(t+b\alpha_{\hat{\mu}}^*)}, C_2^* = g^{ak_i}$, where $r + k_i$ is random in $\mathcal{A}$ perspective. If W is a random element of $\mathbb{G}$, so is C_1^*. In addition, since k_i is a random C_2^* is also random in $\mathcal{A}$ perspective.

(4) More trapdoor queries:

$\mathcal{A}$ sends $\tilde{w}$ to C for more trapdoor queries, where $\tilde{w} \neq w_0^*, \tilde{w} \neq w_1^*$ and C respond to the query as before.

(5) Guess:

$\mathcal{A}$ outputs its guess $\hat{\mu}' \in \{0,1\}$. If $\hat{\mu}' = \hat{\mu}$, then C outputs $\mu' = 0$; otherwise, C outputs $\mu' = 1$.

Refer to [13], we use *ter* to denote two cases in which Challenger C aborts during the game, as follows:

① When C simulates $\mathcal{O}_T$ and $\mathcal{O}_C$, $\tau_i = 0$. Since that each τ_i is picked randomly and independently, the probability that C aborts the game is $\Pr|ter_1| = (1-\theta)^{q_T+q_C}$, where q_T and q_c represent the adversary $\mathcal{A}$ invokes at most q_T, q_C queries to $\mathcal{O}_T$ and $\mathcal{O}_C$, respectively.

② In the challenge keywords chosen by adversary $\mathcal{A}$, $\tau_0^* = \tau_1^* = 1$, the probability that C aborts the game is $\Pr|ter_2| = 1 - (1-\theta)^2$. Hence, the probability of C not aborts in the game is $\Pr\left|\overline{ter}\right| = ((1-\theta)^{q_T+q_C})(1 - (1-\theta)^2)$. When $\theta = 1\text{-}\sqrt{\frac{q_T+q_C}{q_T+q_C+2}}$, the probability $Pr\left|\overline{ter}\right|$ takes the maximum value:

$$Pr\left|\overline{ter}\right| = \left(\frac{q_T + q_C}{q_T + q_C + 2}\right)^{\frac{q_T+q_C}{2}} \cdot \frac{2}{q_T + q_C + 2}, \tag{9}$$

which nearly equal to $\frac{2}{(q_T+q_C)^e}$, and thus non-negligible. Therefore, the success probability that of C guessing the bit μ (i.e., solving the mDLIN problem) is:

$$\Pr|\mu' = \mu| = \frac{1}{2} + \epsilon_C \cdot \Pr\left|\overline{ter}\right|. \tag{10}$$

If ϵ_C is non-negligible, so is $|\Pr[\mu' = \mu] - 1/2|$.

5.2. User's query privacy

Each LBS data will be encrypted and submitted to CS as a query trapdoor when a user query. Specifically, for POI coordinates, as discussed in the encryption phase, the vector $\hat{\boldsymbol{q}}$ will also be split into two random vector $\hat{\boldsymbol{q}}_a$ and $\hat{\boldsymbol{q}}_b$ according to the bit string b. Since the CS cannot obtain the invertible matrix $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and the bit string b, after a series of operations such as splitting and matrix multiplication, CS can not get the user's actual coordinates and query radius with the user's query trapdoor.

Lemma 2. *Allowing the problem of DBDH is hard, then for any PPT $\mathcal{A}$, the probability advantage $\mathrm{Adv}_{\mathcal{A}}^{T}(\lambda)$ who breaches the keywords trapdoor privacy of our scheme is negligible.*

Proof 2. Assuming that there is a probabilistic polynomial time adversary $\mathcal{A}$ can breach the trapdoor privacy of our proposed scheme with a

non-negligible advantage ϵ_T, there is a challenger C whose probability of solving the DBDH problem cannot be negligibled. Given an instance of the DBDH problem with parameters $(g, g^a, g^b, g^c) \in \mathbb{G}, W \in \mathbb{G}_T$, where $a, b, c \in \mathbb{Z}_q^*$, the goal of C is to determine $W = e(g,g)^{abc}$ or a random element in $\mathbb{G}_T$. C sets $\mu \in \{0,1\}$, if $W = e(g,g)^{abc}$, μ=0; if W is a random element from $\mathbb{G}_T$, μ=1.

(1) Initialization:

The C sets $(PK_u, PK_L) = (g^a, g^b)$ and sents it to the adversary along with the public parameter $param = (\mathbb{G}, \mathbb{G}_T, e, q, g)$.

(2) Query:

$\mathcal{A}$ can ask C for the following query:

a) Hash query $\mathcal{O}_H$: $\mathcal{A}$ sends the keyword w_i to C for hash query. C will do the following:

① C maintains an initially empty list $L_H\langle w_i, h_i, \alpha_i, \tau_i\rangle$ for $\mathcal{A}$'s query, and if w_i appears in L_H, C sends a response $H(w_i) = h_i$ to $\mathcal{A}$.

② Otherwise, C randomly chooses $\tau_i \in \{0,1\}$ in probability such that $Pr[\tau_i = 0] = \theta$.

If $\tau_i = 0$, randomly chooses $\alpha_i \in \mathbb{Z}_q^*$ and sets $h_i = g^{z+\alpha_i}$; Otherwise, set $h_i = g^{\alpha_i}$.

③ C stores the tuple $L_H\langle w_i, h_i, \alpha_i, \tau_i\rangle$ to L_H and returns $H(w_i) = h_i$ to $\mathcal{A}$.

b) Ciphertext query $\mathcal{O}_C$: $\mathcal{A}$ sends the keyword w_i to C for ciphertext query and C retrieves $\langle w_i, h_i, \alpha_i, \tau_i\rangle$ from the list L_H. If $\tau_i = 0$, it aborts the query and randomly outputs a bit $\mu' \in \{0,1\}$ as its guess of μ. Otherwise, it randomly picks $k_i \in \mathbb{Z}_q^*$ and computes the ciphertext $C_w = (C_1, C_2) = ((g^{\alpha_i})^b \cdot g^{k_i} = g^{\alpha_i b + k_i}, (g^a)^{k_i})$. C then returns C_w to $\mathcal{A}$.

c) Trapdoor query $\mathcal{O}_T$: $\mathcal{A}$ sends the keyword w_i to C for trapdoor query and C retrieves $\langle w_i, h_i, \alpha_i, \tau_i\rangle$ from L_H. If $\tau_i = 0$, it aborts the query and randomly outputs a bit $\mu' \in \{0,1\}$ as its guess of μ. Otherwise, it randomly picks $r_i \in \mathbb{Z}_q^*$ and derives the trapdoor $T_w = \{T_1, T_2, T_3\} = \{e(g^{\alpha_i a}, g^{br_i}) = e(H(w_i)^a, g^{br_i}), g^{r_i}, g^{ar_i}\}$. C then returns T_w to $\mathcal{A}$.

(3) Challenge:

After polynomial queries, $\mathcal{A}$ sends two keywords w_0^*, w_1^* to C, which have not been queried to $\mathcal{O}_T$ nor $\mathcal{O}_C$. Then, C performs the following operations:

① C performs a Hash query on w_0^*, w_1^* respectively: $H(w_0^*) = h_0^*$ and $H(w_1^*) = h_1^*$. w_0^*, w_1^* correspond to tuples $\langle w_0^*, h_0^*, \alpha_0^*, \tau_0^*\rangle$ and $\langle w_1^*, h_1^*, \alpha_1^*, \tau_1^*\rangle$, respectively. If τ_0^*, τ_1^* are both 1, C aborts the query and randomly outputs a bit $\mu' \in \{0,1\}$as its guess of μ;

② If at least one of τ_0^* and τ_1^* is 0, let $\hat{\mu}$ be the bit satisfying $\tau_{\hat{\mu}}^* = 0$. C calculates the trapdoor $T^* = \{T_1^*, T_2^*, T_3^*\}$, where $T_1^* = W \cdot e(g^a, g^{r_i b})^{\alpha_{\hat{\mu}}^*}, T_2^* = g^{r_i}, T_2^* = g^{ar_i}$. If $W = e(g,g)^{abc}$, then $T_1^* = e(g,g)^{ab(c+r_i\alpha_{\hat{\mu}}^*)}$. Since $r_i\alpha_{\hat{\mu}}^*$ is a random value, so $T_1^* = e(g,g)^{ab(c+r_i\alpha_{\hat{\mu}}^*)} = e(h_{\hat{\mu}'}, g^{ab})$, C outputs 0; if W is a random member of $\mathbb{G}_T$, then T_1^* is also a random element in $\mathbb{G}_T$, C outputs 1. In addition, since r_i is a random value, T_2^*, T_3^* are also random for $\mathcal{A}$.

(4) More trapdoor queries:

$\mathcal{A}$ sends $\tilde{w}$ to C for more trapdoor queries, where $\tilde{w} \neq w_0^*, \tilde{w} \neq w_1^*$ and C respond to the query as before.

(5) Guess:

$\mathcal{A}$ outputs its guess $\hat{\mu}' \in \{0,1\}$, if $\hat{\mu}' = \hat{\mu}$, then C outputs $\mu' = 0$, otherwise C outputs $\mu' = 1$.

Refer to [13], we use *ter* to denote two cases in which Challenger C aborts during the game, as follows:

① When C simulates $\mathcal{O}_T$ and $\mathcal{O}_C$, $\tau_i = 0$. Since that each τ_i is picked randomly and separately, the probability that C aborts the game is $\Pr|ter_1| = (1-\theta)^{q_T+q_C}$, where q_T and q_c represent the adversary $\mathcal{A}$ invokes at most q_T, q_C queries to $\mathcal{O}_T$ and $\mathcal{O}_C$, respectively.

② In the challenge keywords chosen by adversary $\mathcal{A}$, $\tau_0^* = \tau_1^* = 1$, the probability that C aborts the game is $\Pr|ter_2| = 1-(1-\theta)^2$. Hence, the probability of C not aborts in the game is $\Pr|\overline{ter}| = ((1-\theta)^{q_T+q_C})(1-(1-\theta)^2)$. When $\theta = 1-\sqrt{\frac{q_T+q_C}{q_T+q_C+2}}$, the probability $Pr|\overline{ter}|$ takes the maximum value:

$$Pr|\overline{ter}| = \left(\frac{q_T+q_C}{q_T+q_C+2}\right)^{\frac{q_T+q_C}{2}} \cdot \frac{2}{q_T+q_C+2}, \tag{11}$$

which nearly equal to $\frac{2}{(q_T+q_C)^e}$, and thus non-negligible. Therefore, the success probability that of C guessing the bit μ (i.e., solving the DBDH problem) is:

$$\Pr|\mu' = \mu| = \frac{1}{2} + \epsilon_T \cdot \Pr|\overline{ter}|. \tag{12}$$

If ϵ_T is non-negligible, so is $|\Pr[\mu' = \mu] - 1/2|$.

Table 2
Symbol definition.

Symbol	Definition
N	The amount of LBS data
n	The data dimension, this paper assumes that n=2
T_e	The running time of one exponentiation operation over the group
T_m	The running time of one multiplication operation over the group
T_p	The running time of a bilinear pairing operation
T_{om}	The running time of a normal multiplication operation

5.3. Index security

Lemma 3. *Even if an adversary $\mathcal{A}$ obtains the index, the probability of the $\mathcal{A}$ successfully deriving the keyword or coordinates is negligible.*

Proof 3. In our scheme, the bit string of the subregion is stored in each non-leaf node of the coded quadtree, and the LBS data ciphertext is stored in the leaf node. Even if the $\mathcal{A}$ obtains the coded quadtree, the data ciphertext cannot be decrypted without the key. When a user's query, it is assumed that the query range is $\{(x_l, y_l), (x_r, y_r), n\}$, where (x_l, y_l) and (x_r, y_r) represent the top left and bottom right vertices of query region Q respectively, and n denotes the amount of subregions in query region Q. Since the user's position coordinates can be in any of the divided subregions, the probability of correctly guessing the user's location in a certain subregion is 1/n. In addition, in view of the one-wayness of the hash function, even if the index $\boldsymbol{V}_{BF}$ is leaked, the adversary can not obtain any valuable information.

6. Performance evaluation

In this section, we analyze the performance of PPT-LBS from the perspectives of data encryption, trapdoor generation and data retrieval, respectively. The experiments are run in Windows 10 operating system with Intel (R) Core (TM) i5-10200H CPU @2.40GHz and 16GB memory. The cryptographic operations have been realized by using the Java Pairing-Based Cryptography (JPBC) library.

(1) Comparison of computational cost

For convenience, Table 2 lists the symbol description definitions used in the comparison. Since the time cost of matrix and vector multiplication operation, hashing operation, symmetric encryption and decryption operation is relatively low, it is ignored in the comparison.

Table 3 shows the comparison results of PPT-LBS and the existing schemes [9,14,15] with regard to computational cost.

(2) Comparison of characteristics

In Table 4, characteristics of our scheme and the existing schemes [9,14,15] are compared.

(3) Comparison of experiments

Figure 9 indicates the comparison results of our scheme and the existing similar scheme [9,14,15] in the data encryption phase. Among them, our scheme and Ou et al.'s [14] scheme use matrix operation and public key searchable encryption to process LBS coordinates and keyword. The computational cost is lower than Lin et al.'s [15] scheme and Zhu et al.'s [9] scheme, which uses a homomorphic encryption

Table 3
Comparison of computational cost in each phase.

	Data encryption	Trapdoor generation	Data retrieval
Ou et al. [14]	$N(T_p + T_m)$	$2T_e$	$N(2T_p + T_m)$
Zhu et al. [9]	$N(6T_e + 2T_m)$	$4NT_e$	$log_2 N \cdot (5T_m + 2T_p)$
Lin et al. [15]	$Nn(3T_e + T_{om} + T_m)$	$2n(T_e + T_{om})$	$log_2 N \cdot (nT_p + (2n+1)T_m)$
Ours	$3NT_e$	$2T_e + T_p$	$log_4 N \cdot (2T_p + T_m)$

Table 4
Comparison of characteristics.

	Query privacy	Location privacy	Geographic range query	Keyword query	Top-k query
Ou et al. [14]	√	√	√	√	×
Zhu et al. [9]	√	√	√	×	×
Lin et al. [15]	√	√	√	×	×
Ours	√	√	√	√	√

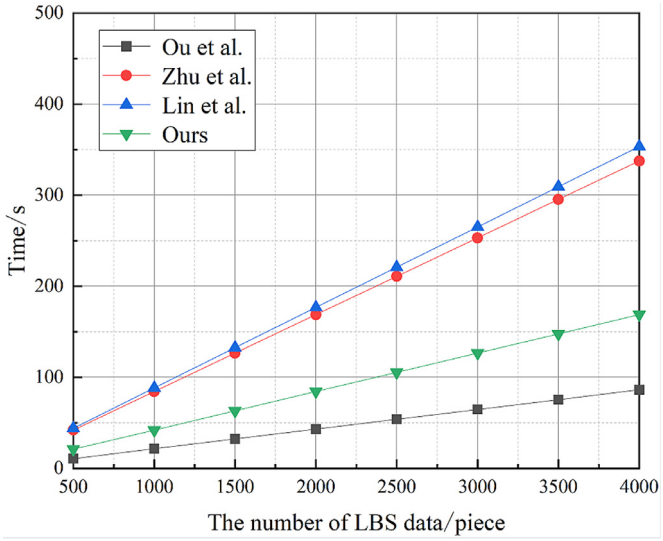


Fig. 9. The experiment comparison of LSP data encryption.

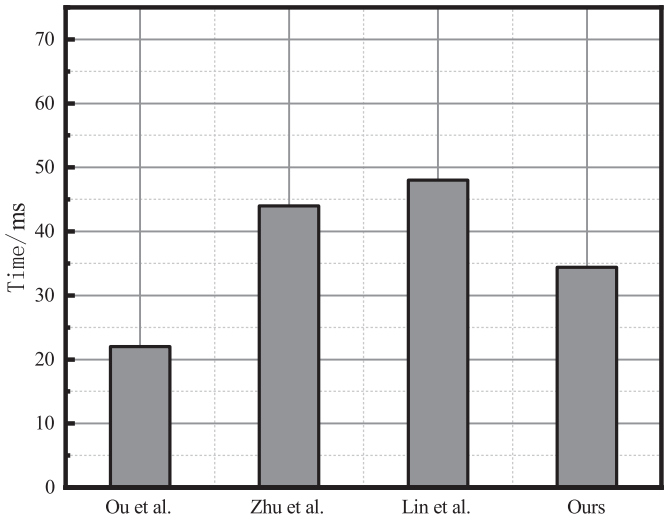


Fig. 10. The experiment comparison of user's trapdoor generation.

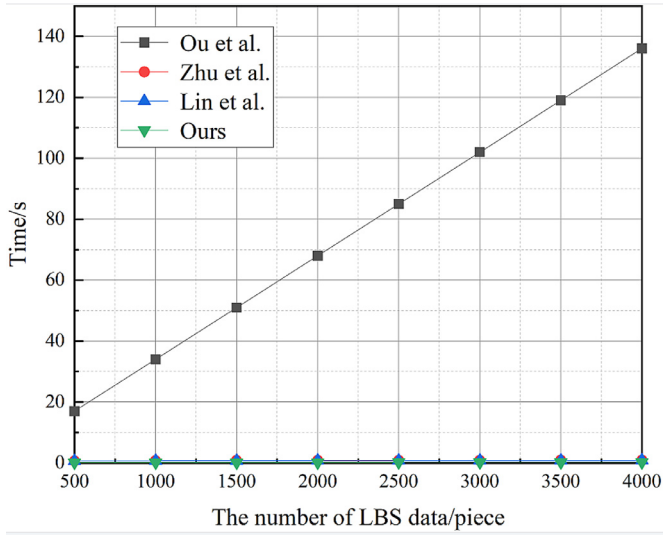


Fig. 11. The experiment comparison of the data retrieval.

mechanism over the composite order group. Lin et al.'s scheme extends data to n dimension on the basis of Zhu et al.'s scheme, so it has the highest cost. Although Ou et al.'s scheme's calculation cost is lower than ours, the key matrix of Ou et al.'s scheme is only 3×3 dimensions, which is easy to be cracked by force and has certain security risks.

Figure 10 shows the comparison results of ours scheme and the existing similar scheme [9,14,15] in the trapdoor generation phase. Similar to the encryption process, our scheme has a lower cost than the Zhu et al.'s [9] scheme and Lin et al.'s [15] scheme using a homomorphic encryption mechanism. Besides, in Zhu et al.'s scheme and Lin et al.'s scheme, users do not support constructing a query trapdoor based on a keyword but can only return all the data in the query area through the CS to decrypt and filter the required data. In the Ou et al.'s [14] scheme, when a user query the same keyword, the same parameters will be generated in the trapdoor, which is not random and easy to disclose the user query mode, resulting in the disclosure of user privacy.

Figure 11 shows the comparison results of ours scheme and the existing similar scheme [9,14,15] in the data retrieval phase. Zhu et al.'s [9] scheme and Lin et al.'s [15] scheme use dichotomy to retrieve data, our scheme uses coded quadtree to retrieve data, the time complexity are both $O(log N)$. Still, neither Zhu et al.'s scheme nor Lin et al.'s scheme supports distance top-k query, while Ou et al.'s [14] scheme need to traverse all of the data and time complexity is $O(N)$, which cannot be applied to massive data.

Therefore, comprehensively, our scheme is suitable for LBS query in outsourcing environment.

7. Conclusion

A privacy-preserving top-k query scheme for outsourcing situation is constructed based on enhanced asymmetric scalar-product preserving encryption and public key searchable encryption in this paper. In addition, we use the coded quadtree and bloom filter to construct an index structure, which enables the CS fast locate the user's query area in the massive encrypted data. Security analysis demonstrates that our scheme not only ensures the data security of LSP but also protects the privacy of users' query, and has preferable to similar scheme in performance.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

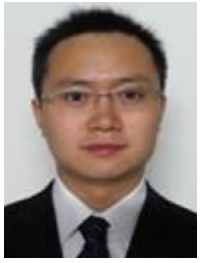
Our work was jointly supported by the National Natural Science Foundation of China (No. 61872051, No. 61702067), the Chongqing Natural Science Foundation of China (No. cstc2020jcyj-msxmX0343), and the Venture & Innovation Support Program for Chongqing Overseas Returnees (No. CX2018122).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejps.2020.105216.

References

[1] X. Zhang, X. Gui, Z. Wu, Privacy preservation for location-based services: a survey, J. Softw. 26 (9) (2015) 2373–2395.
[2] X. Zhu, E. Ayday, R. Vitenberg, A privacy-preserving framework for outsourcing location-based services to the cloud, IEEE Trans. Dependable Secure Comput. (2019).
[3] Z. Liu, L. Wu, J. Ke, W. Qu, W. Wang, H. Wang, Accountable outsourcing location-based services with privacy preservation, IEEE Access 7 (2019) 117258–117273.
[4] R. Li, A.X. Liu, A.L. Wang, B. Bruhadeshwar, Fast and scalable range query processing with strong privacy protection for cloud computing, IEEE/ACM Trans. Netw. 24 (4) (2015) 2305–2318.
[5] R. Li, A.X. Liu, Adaptively secure conjunctive query processing over encrypted data for cloud computing, in: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), IEEE, 2017, pp. 697–708.
[6] M. Zeng, K. Zhang, J. Chen, H. Qian, P3GQ: a practical privacy-preserving generic location-based services query scheme, Pervasive Mob. Comput. 51 (2018) 56–72.
[7] S. Yang, S. Tang, X. Zhang, Privacy-preserving k nearest neighbor query with authentication on road networks, J. Parallel Distrib. Comput. 134 (2019) 25–36.
[8] Q. Xie, L. Wang, Privacy-preserving location-based service scheme for mobile sensing data, Sensors 16 (12) (2016) 1993.
[9] H. Zhu, R. Lu, C. Huang, L. Chen, H. Li, An efficient privacy-preserving location-based services query scheme in outsourced cloud, IEEE Trans. Veh. Technol. 65 (9) (2015) 7729–7739.
[10] H. Zhu, F. Liu, H. Li, Efficient and privacy-preserving polygons spatial query framework for location-based services, IEEE Internet Things J. 4 (2) (2016) 536–545.
[11] L. Zhao, Q. Liu, H. Huang, X. Jia, Efficient privacy-preserving query processing on outsourced geographic databases, in: 2018 IEEE Global Communications Conference (GLOBECOM), IEEE, 2018, pp. 1–6.
[12] W.K. Wong, D.W.-l. Cheung, B. Kao, N. Mamoulis, Secure kNN computation on encrypted databases, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, 2009, pp. 139–152.
[13] Q. Huang, H. Li, An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks, Inf. Sci. 403 (2017) 1–14.
[14] L. Ou, H. Yin, Z. Qin, S. Xiao, G. Yang, Y. Hu, An efficient and privacy-preserving multiuser cloud-based LBS query scheme, Secur. Commun. Netw. 2018 (2018).
[15] J. Lin, J. Niu, H. Li, M. Atiquzzaman, A secure and efficient location-based service scheme for smart transportation, Future Gener. Comput. Syst. 92 (2019) 694–704.



Zhou Yousheng, received the PhD degree from the Beijing University of Posts and Telecommunications, in 2011. He is currently an associate professor with the Chongqing University of Posts and Telecommunications. He has published more than 20 academic papers in peer-reviewed international journals. His research interests include mobile security and cloud security. Email: zhouys@cqupt.edu.cn

Li Xia, is currently a master student of School of Cyber Security and Information Law, Chongqing University of Posts and Telecommunications. Her research interest include blockchain security and the IoT security. Email: lixiacr7@163.com

Wang Ming, received the master' s degree from the Chongqing University of Posts and Telecommunications. His research interests include mobile security and the IoT security. Email: s180231886@stu.cqupt.edu.cn

Liu Yuanni, is an associate professor at the Institute of Future Network Technologies, ChongQing University of Posts and Telecommunications, China. She received her PhD from the Department of Network Technology Institute, Beijing University of Posts and Telecommunications, China, in 2011. Her research interests include mobile crowd sensing, IoT security, and data virtualization. Email: liuyn@cqupt.edu.cn