Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

An Efficient Privacy-Preserving Multi-keyword Query Scheme in Location Based Services

SHIWEN ZHANG[1,2], TINGTING YAO[3], WEI LIANG [4], VOUNDI KOE ARTHUR SANDOR[4], AND KUAN-CHING LI[5], (Senior Member, IEEE)

[1]College of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, Hunan, 411201, China
[2]College of Computer, National University of Defense Technology, Changsha, Hunan, 410073, China
[3]Hunan Provincial Key Laboratory of Network Investigational Technology, Hunan Police Academy, Changsha, Hunan, 410138, China
[4]College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, 410082, China
[5]Department of Computer Science and Information Engineering, Providence University, Taichung, Taiwan

Corresponding author: Shiwen Zhang (e-mail: shiwenzhang@hnust.edu.cn), Wei Liang (e-mail:weiliang99@hnu.edu.cn).

This work is supported in part by the National Natural Science Foundation of China (No.61702180), the Natural Science Foundation of Hunan Province (No.2019JJ50167), the Doctoral Scientific Research Foundation of Hunan University of Science and Technology (No.E51790), the Open Research Fund of the Hunan Provincial Key Laboratory of Network Investigational Technology (No.2017WLZC008), and the General Project Research Foundation of Education Bureau of Hunan Province (No.18C0345).

ABSTRACT With the proliferation of location-aware mobile devices and the prevalence of wireless communications, location-based services (LBS) have attracted much particular attention in recent years. For flexibility and cost savings, the LBS provider outsources the LBS data to the cloud in order to serve the increasing number of mobile users. To guarantee users' privacy and data confidentiality, some excellent works have been proposed which focus on secure query over the location server. However, these existing works have two limitations. On the one hand, they cannot preserve users' location and query content privacy simultaneously. On the other hand, they fail to support multi-keyword queries. In this paper, aiming at a multi-keywords query in LBS, we propose a novel efficient and privacy-preserving multi-keyword query scheme (PPMQ) over the outsourced cloud, which satisfies the requirements of the location and query content privacy protection, query efficiency, the confidentiality of LBS data and scalability regarding the data users. To improve the efficiency of our proposed scheme, we utilize the linear quad-tree technique to build a grid system to represent the location information in the query condition as well as a searchable index. To protect the location privacy, we combine decimal Morton code and public-key cryptography techniques to build a searchable index or to generate a trapdoor. To enable the cloud server to perform a secure multi-keyword query, we systematically construct a privacy-preserving query scheme with bilinear pairing-based cryptography. In particular, our proposed scheme is scalable and very suitable for multi-user environments due to the flexible user registration and revocation mechanisms. Furthermore, a detailed security analysis shows that the proposed scheme can ensure the confidentiality of LBS data, and protect the location and query content privacy. Extensive experiments are conducted on a real LBS dataset, and the simulation results confirm the security and efficiency of our scheme.

INDEX TERMS Multi-keyword query, Location-based services, Linear quad-tree, Bilinear pairing map

I. INTRODUCTION

WITH the proliferation of wireless communication technologies and mobile devices with positioning capabilities, location-based services (LBS) are being extensively used in our daily life. LBS can help people enjoying a comfortable life. It is reported that more than 150 million people have enjoyed LBS in 2014 [1]. The most common and typical service of the LBS system is location query service, which has been applied to many areas, such as transportation, target advertising, friend recommendation, a restaurant finder, and so on [2, 3]. The location query allows a user to search nearby points of interest (POIs) within a given distance to her/his location. Then, the LBS provider returns the desirable POIs to the query users.

Although LBS can improve people's life more conveniently, its large adoption still faces severe challenges. The information security and privacy preservation problems, especially those regarding user's location and query content privacy, should be addressed before the deployment of LBS in the real-world [4, 5]. In a LBS system, the data queriers first submit their accurate locations and query contents to the LBS provider, then the LBS provider returns the desirable POIs records to the data queries. By collecting and analyzing query users' current location and query content, the LBS provider can easily obtain a great deal of sensitive information, such as users' real identities, health status, trade secrets, hobbies, and so on on [6]. At the same time, the querying user would only be interested in a few POIs, which are more relevant to her/his query. Returning a larger number of POIs will cause considerable computation and communication costs. Hence, designing a privacy-preserving efficient query scheme in LBS systems that protects the user's location and the query content privacy is still an active topic of LBS research.

Let us consider the following application scenario. To achieve low computation cost and flexible LBS deployment, the LBS provider can be regarded as a data owner, which outsources his/her LBS data (i.e., POIs) to a cloud server for enjoying the abundant benefits brought by cloud computing such as easy access, great flexibility, cost-saving, excellent computation performance, and others. For the sake of protecting the sensitive LBS data, all data are encrypted before being outsourced to the cloud server. The authorized data users, i.e., registered users, can issue a multi-keyword query to find desirable POIs records within a given distance to his/her current location from the cloud server. Then, the cloud server searches its database and returns the corresponding POIs to the data user. Such results help the data user to look for desirable POIs accurately and quickly. For example, a tourist can issue the following multi-keyword query to the LBS provider through his/her smartphone: " what is the lowest price for the most popular hotel within 1000 meters from my current location ?". Then, the cloud server searches for the POIs at a given distance and returns the charming hotels considering price and reviews.

To protect the privacy of location data in the LBS system, scholars have dedicated many research efforts to design privacy preserving schemes for LBS [7]. Many solutions have been proposed in the literature [6, 8–24]. However, most of schemes only protect either the location privacy [9–11] or the query content privacy [14]. They fail to preserve the location privacy and query content privacy simultaneously. In [12, 13, 15–17], many approaches have been proposed to protect the location information and the query content privacy. But, they downgrade the accuracy of the user's location information and increase the communication cost between the user and the LBS provider. Encryption based on the various schemes [2, 4, 18, 19, 21] can fully ensure the security of data and provide accurate results to users. However, most of those techniques bring relatively high computation or communication costs on the user side, which lead to much energy consumption over the mobile devices. Furthermore, most of the prior works either address privacy preserving, or instead result in poor user experience on POIs query. For instance, such works only support location coordinate query or single keyword search. However, nowadays, users prefer to submit multiple keywords to retrieve the most relevant POIs. Therefore, it is challenging to develop a secure, efficient multi-keyword query scheme over encrypted LBS location data in a flexible and scalable manner.

In this paper, different from existing works, aiming at the challenges as mentioned above, we propose an efficient privacy-preserving multi-keyword query scheme in LBS, named PPMQ, which provides fine-grained queries and returns more accurate POIs to users without divulging users' sensitive information to both the cloud server or to other unregistered users. To prevent the cloud server and the unauthorized users from knowing the exact location data of the data owner, we adopt a systematic encryption to encrypt the outsourced LBS data. To protect the location and query content privacy, we first utilize linear quad-tree to design a perfect grid system, in which the real location information of the user and POI record can be represented as a grid location coordinate. Then, combined quad-tree with decimal Morton code technology, a secure index construction and trapdoor generation algorithm is developed. To enable the cloud server to perform secure multi-keyword searches, based on bilinear pairing cryptography, we construct a secure query protocol. Users can issue a multi-keyword query to accurately locate the desirable POIs quickly and conveniently without revealing any sensitive information, which dramatically improves the user experience. Furthermore, PPMQ is very suitable for multi-user environments by providing flexible registration and revocation mechanisms for users, which can help the data owner to carry out the user's identity authentication. Finally, we conduct extensive experiments on real-world LBS datasets and give a rigorous security analysis to confirm the proposed scheme's security and efficiency.

To summarize, the main contributions of this paper are:

- We propose an efficient and privacy-preserving multi-keyword query scheme that can simultaneously preserve the location and query content privacy. Besides, with flexible user registration and revocation mechanisms, our scheme is very suitable for the multi-user environment.
- We systematically construct a secure multi-keyword query protocol, which not only enables the cloud servers to perform a secure multi-keyword search without knowing the actual value of both query condition and POIs but also allows the data owners to encrypt keywords of POIs with their secret keys, such that the registered data users can query the POIs without knowledge of any secret key.
- To achieve query efficiency, we first utilize the quadtree technique to build a grid system representing the location information of user and POIs. Then, combining with the Morton coding algorithm, we can build a searchable index and generate a trapdoor securely. We

develop a fine-grained query protocol, where the data user can query the POIs by initiating a multi-keyword query and obtaining the desirable POIs according to their preference.

- We give rigorous security analysis and conduct extensive experiments on a real LBS data set. The analysis shows that our scheme can protect user location privacy and guarantee the confidentiality of LBS data simultaneously. Experimental results confirm the efficiency and effectiveness of our proposed scheme.

The rest of our paper is structured as follows. We review some related works in Section II. In Section III, we formalize the system model, threat model and design goal. Then, the preliminaries are presented in Section IV. In Section V, we provide the location representation model and define the proposed scheme. Moreover, our construction of the PPMQ scheme is presented in Section VI, followed by the security analysis of our scheme. The performance evaluations are conducted in Sections VII and VIII, respectively. Finally, we conclude our paper in Section IX.

II. RELATED WORK

A. LOCATION PRIVACY PRESERVATION IN LOCATION BASED SERVICE

Protecting user's location privacy in LBS has drawn a lot of attention from researchers in recent years. Existing privacy-preserving techniques in the LBS ecosystem can be broadly categorized into four groups [7]: Anonymity based schemes [9–11], Obfuscation mechanisms [12–17], Encryption based schemes [18–20], and shared information reduction mechanisms [21–23]. We briefly discuss some of them as follows.

Anonymity based schemes aim to break the links between users' identity and location information, such as k-anonymity [9], l-diversity [10] and p-sensitivity [11]. These schemes can protect the user's identity and location privacy effectively. However, they need a trusted third party (TTP) to blur users' exact location information into a cloaked area. The TTP would easily become the target of attacks and exhibit a single point of failure vulnerability. To avoid using TTP, obfuscation based schemes reduce the precision of users' location information by adding dummy locations [12] or noise [15, 16] and generalizing location data [13, 14]. Nevertheless, most of the schemes introduce additional system costs or sacrifice the utility of the location data. Encryption based schemes, such as homomorphic encryption schemes [17, 18, 20] can provide LBS accurately while protecting the confidentiality of users' location data. Nevertheless, most of them impose relatively high computation requirements on the user side, which is not suitable for resource-constraint mobile devices.

Unlike existing works, we propose a privacy-preserving multi-keyword query protocol in LBS, which enables the data user to obtain the desirable POIs more accurately and more conveniently without divulging the location and the query content privacy. The proposed scheme is flexible and efficient.

B. SECURE KEYWORD SEARCH

To protect the sensitive information of users and enable the cloud server to perform a keyword search, Wang et al. [25] defined and solved the secure ranked keyword search over encrypted cloud data. In [25], they found that invariably retrieving all files and returning undifferentiated results would incur considerable communication costs for the data querier to get the most relevant files. They propose a secure keyword search scheme that returns top-k relevant files upon a single keyword based on an order-preserving symmetric encryption technique. To further enhance search efficiency, Curtmola et al. [26] developed a single encrypted hash table index for the entire file collection and then proposed a per-keyword based scheme. However, as a common practice indicated by today's web search engines (e.g., Google search), data users tend to issue a multiple keywords search rather than single keyword search to retrieve the most relevant data. Further, In [27], based on secure inner product computation, Cao et al. proposed a preserving multi-keyword ranked search over encrypted cloud data (MRSE). [28] also extended the secure keyword search for multi-keyword queries. Their approaches employ "inner product similarity" to quantitatively evaluate the similarity between query keywords and outsourced data files. Zhang et.al [29] proposed a scheme that deals with secure ranked multi-keyword search in a multi-owners model. To tolerate both of minor typos and format inconsistencies given user's search input, Li et al. [30] and Chuah et al. [31] proposed fuzzy keyword search over encrypted data. To enrich query predicates, conjunctive keyword search [32, 33] over encrypted data has also been proposed. As a more general search approach, predicate encryption schemes [34, 34, 36] are recently proposed to support both conjunctive and disjunctive search.

Motivated by multi-keyword search in could computing, we focus on the secure multi-keyword query in LBS, enabling the LBS provider to return the most relevant POIs to query users accurately. Also, we seek to achieve an efficient and scalable query system without sacrificing the user's privacy and security of data, so that the scheme could be suitable for a more significant number of data users.

III. PROBLEM FORMULATION

A. SYSTEM MODEL

First, we describe the notations in this paper, as shown in Table. 1. Then, we present the system model. In Fig. 1, our system model consists of three different entities: the data owner, the cloud server and the multiple data users. The data owner (i.e., LBS provider) collects a series of location and related information from the business called poi records so as to provide LBS service to data users. These poi records comprise a LBS location dataset. Since the cloud server can provide low-cost storage and powerful computation services, we assume that the data owner is willing to outsource the

TABLE 1. Summary of Notations

Notations	Descriptions
$ID_i, loc_i = (lat_i, lon_i)$	The identifier, and the location latitude and longitude coordinates of the poi_i
$W_i = \{w_{i1}, w_{i2}, \cdots, w_{im}\}$	The set of keywords that describes the poi_i in all aspect
$poi_i = (ID_i, loc_i, W_i)$	The point of interest record poi_i
$\mathcal{POI} = \{poi_1, poi_2, \ldots, poi_n\}$	The POIs set
$\widetilde{\mathcal{POI}} = \{\widetilde{poi_1}, \widetilde{poi_2}, \cdots, \widetilde{poi_n}\}$	The encrypted POIs set
$\widetilde{\mathcal{W}} = \{\widetilde{W_1}, \widetilde{W_2}, \cdots, \widetilde{W_n}\}$	The encrypted keywords set associated with $\mathcal{POI}$, where each $\widetilde{W_i}$ is the encrypted form of W_i
$\mathcal{I} = (I_1, I_2, \cdots, I_n)$	The searchable index associated with $\mathcal{POI}$, where each subindex I_i is built for poi_i
M_i, QM	The Morton code value of the poi_i and the set of Morton code values of query region
$\widehat{\mathcal{W}} = (w_1, w_2, \cdots, w_t), T_{\widehat{\mathcal{W}}}$	The query keywords, and the trapdoor for the query keywords $\widehat{\mathcal{W}}$
$\widetilde{\mathcal{POI}}_{\widehat{\mathcal{W}}}$	The relevant encrypted POIs that contains query keywords $\widehat{\mathcal{W}}$
UL, ERK_u	The list of identifiers of registered users, and the search public key of the user u
$\widetilde{M_i}, KF(\cdot)$	The encrypted Morton code value for the point poi_i, and a key derivation function
$\widetilde{QM} = (\widetilde{M_1}, \widetilde{M_2}, \cdots, \widetilde{M_\mu})$	The encrypted Morton code values set of the query region

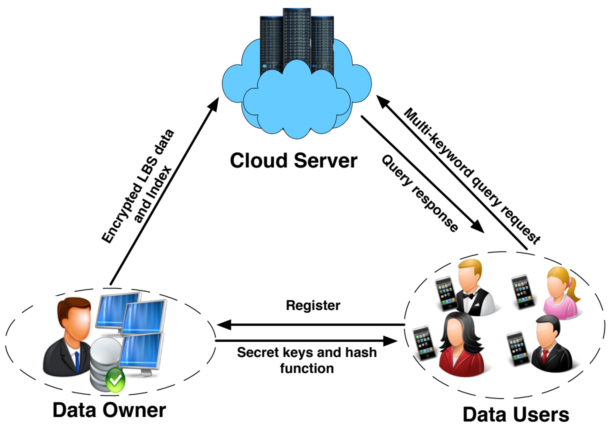

Cloud Server
Encrypted LBS data and Index
Multi-keyword query request
Query response
Register
Secret keys and hash function
Data Owner
Data Users

FIGURE 1. System model for proposed scheme.

location dataset to the cloud server for better LBS offers to data users. In order to protect the confidentially and privacy of the location data, the data owner encrypts each poi_i record before outsourcing it to the cloud server. When a data user wants to join the system, the LBS provider provides authentication and registration service for the data user. The data users must provide their own identity information to register with the data owner (i.e., LBS provider). If the data user passes the authentication, the data owner grants the search capabilities to the legal users by sending some important security parameters to data users. The legal data users can enjoy the LBS service by submitting their multi-keyword queries to the cloud server. After that, the cloud checks the identity information and search capabilities of data users and then performs the query process. Without the valid security parameters of the data users, the cloud cannot complete the query process for the data users. At last, the cloud returns query results to the data users. Correspondingly, the LBS provider can also revoke any expired data user, who no longer has the search capability over the outsourced LBS data. Note that how to achieve decryption capabilities is out of the scope of this paper, some excellent work to this problem can be found in [36, 37]. Attribute based encryption is a better way to manage user's access towards outsourced LBS data. Next, we will describe each entity of our model as follows in detail.

1) **Data Owner**: Data owner assumes the role of a LBS provider (LBSP) who owns a large-scale poi records, denoted as $\mathcal{POI} = \{poi_1, poi_2, \ldots, poi_n\}$. For convenience, we assume each record poi_i contains three elements. We use (ID_i, loc_i, W_i) to represent the poi_i record, where ID_i is the identifier of the poi_i's record, $loc_i = (lat_i, lon_i)$ denotes the location coordinate, $W_i = \{w_{i1}, w_{i2}, \cdots, w_{im}\}$ is the skeleton description of the poi_i record containing m keywords. To prevent the cloud server from knowing the actual content of the poi record, the data owner encrypts each poi_i record to form an encrypted location dataset, denoted as $\widetilde{\mathcal{POI}} = \{\widetilde{poi_1}, \widetilde{poi_2}, \ldots, \widetilde{poi_n}\}$. To enable efficient search on the encrypted LBS location data $\widetilde{\mathcal{POI}}$, the data owner has to build a secure searchable index $\mathcal{I}$ for the location data. Finally, both searchable index $\mathcal{I}$ and $\widetilde{\mathcal{POI}}$ are outsourced to the cloud server.
2) **Cloud Server**: Cloud server stores the encrypted LBS data $\widetilde{\mathcal{POI}}$ and verifies the search capabilities for the data user. If the data user obtained the search capabilities from the data owner, then the cloud server stores the search public key and identity information of the registered data user. Thus, the registered data user can pass the authentication for being a valid user. When receiving the encrypted queries from the authorized users, the cloud server performs the secure multi-

keyword search over encrypted data $\widetilde{\mathcal{POI}}$ and then returns the satisfied query results to the users. The cloud server does not know any context of the poi_i records, the user's query context, or the location of authorized users. Only the authorized users can search the encrypted dataset and recover the query results after sending a multi-keyword query to the cloud.

3) **Data Users**: The data users are authorized LBS users, who enjoy convenient location-based services by submitting multi-keyword queries to the cloud server anywhere and anytime. The data user first registers him/herself with the data owner and then obtains a secret key and a secret hash function from the data owner. For example, the query request, "find a 24-hour Indian curry restaurant near me", contains three keywords, "24-hour", "Indian curry" and "restaurant". To hide the query request for protecting query privacy, the user can use his secret key to encrypt the query keywords and then send the encrypted query keywords to the cloud server. When a data user receives the query results returned from the cloud server, he/she can recover the actual content of the POIs by decryption. The unregistered and revoked users from the data owner cannot enjoy the LBS query service.

B. THREAT MODEL

In this paper, we mainly consider two attacks, the external attacks, and the internal attacks. Unauthorized outsiders initiate external attacks. We can build secure communication channels between all parts using standard security protocols, such as Secure Socket Layer (SSL) [37] and Secure Socket Shell (SSH) protocol [38], to resist the external attacks. The SSL and SSH protocol can use a combination of cryptographic processes to provide secure access to a computer over an unsecured network. Thus, we only focus on the internal attacks initiated by the cloud server and the unregistered data users. In our threat model, we assume that the data owner and authorized data users are trusted. However, the cloud server is not trusted. We regard it as "honest but curious", which is the same as previous works [25, 27, 36, 39]. That is to say, the cloud server is "curious" to learn and infer the encrypted record $\widetilde{poi_i}$ and the received message. It may attempt to deduce the actual information of the user's query content, stored LBS data, and location information of users. We also assume that the cloud cannot collude with revoked users to derive additional information about the data owner's encrypted poi records.

C. DESIGN GOAL

1) **Security Guarantee**: The proposed scheme should prevent the cloud server from inferring the accurate location of users, user's query content, and the actual contents and keywords value of encrypted poi records stored in the cloud. The cloud server should not know the user's query interest, the exact location of users, and the context of each encrypted poi record. Besides, we still need to guarantee that the cloud server cannot recover the actual content of the query result.
2) **Access Control**: The cloud server only provides multi-keyword query service to current data users who have been authorized by the data owner (i.e., LBSP). The unregistered or revoked users cannot enjoy this LBS service.
3) **Scalability**: This system can provide a multi-keyword query service for a large number of data users at the same time. The proposed scheme allows a data user to enter or leave the system without affecting other data users.
4) **Computation Efficiency**: The cloud server should process the multi-keyword query efficiently without disclosing query content and location privacy of data users. The data owner should encrypt these poi records speedily and then send it to the cloud. Moreover, the data users also can compute the trapdoors quickly according to the query condition. The proposed scheme should be as efficient as possible.

IV. PRELIMINARIES

In this section, we recall the bilinear paring map and review the quad-tree technique, which will serve as the basis of our proposed PPMQ scheme.

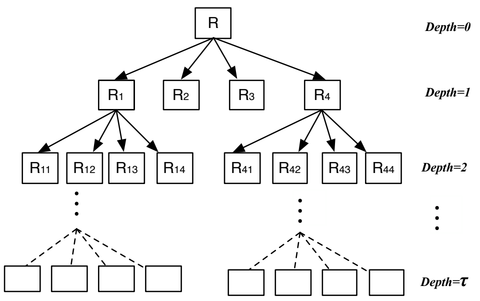
A. BILINEAR PAIRING MAP

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups with the same large prime order q, and let g be a generator of $\mathbb{G}$. A bilinear pairing map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ has the following properties: 1) Bilinearity, i.e., for all $a, b \in \mathbb{Z}_q^*$ and $u, v \in \mathbb{G}$, we have $e(u^a, v^b) = e(u, v)^{ab}$; 2) Non-degeneracy, if $g \in \mathbb{G}$, then $e(g, g) \neq 1 \in \mathbb{G}_T$; 3) Computability, for all $u, v \in \mathbb{G}$, there exists a efficient algorithm to compute $e(u, v) \in \mathbb{G}_T$.

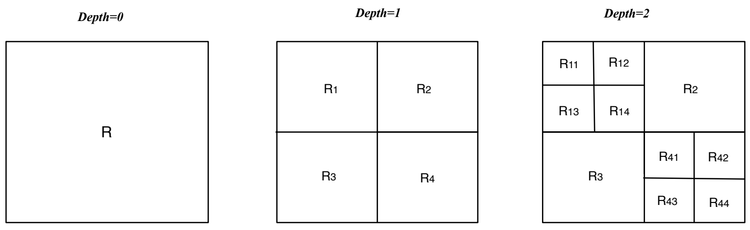
Definition 1 (Discrete logarithm problem:): Given a multiplicative cyclic group $\mathbb{G}$ with the prime order q, g is a generator of $\mathbb{G}$, we first select an element a from $\mathbb{Z}_q^*$, and compute $g^a \in \mathbb{G}$. Then, it is difficult to compute the correct value of a. In other words, given a tuple $(\mathbb{G}, q, g^a, g)$, there is not an efficient algorithm to output a.

B. QUADTREE STRUCTURE

A quadtree is a tree data structure in which each internal node has exactly four children, as shown in Fig. 2. Quadtrees is most often used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions [40]. Quadtrees are used to store data of point on a two-dimensional space efficiently. For example, a quadtree provides a uniform space decomposition mechanism for spatial data such as coordinates in a Geographic Information System (GIS). Quadtree decomposes the location coordinate space into a hierarchy tree. Due to its simplicity and regularity, the quadtree technique has been widely applied in many applications, such as image compression, collision detection, search for nearby points.



(a) Quadtree



(b) The Grid System Corresponding to the Above Quadtree

FIGURE 2. The structure of the quadtree.

The process of constructing a linear quadtree from a two-dimensional spatial area is described as follows. First of all, we assume that the depth of a quadtree has a maximum value. Then, we recursively divide a city area on the map into four equally-sized parts, forming four grids with the same sizes. If the depth of the quadtree does not reach the maximum value, the grid is split into four smaller grids with the same size, and the POIs in the parent grid is inserted into child grids. In this way, a hierarchy quadtree has been built from this region. As we can see from Fig. 2, if the city area has been recursively divided τ times, the depth of quadtree is τ. This quadtree has $2^\tau \times 2^\tau$ leaf nodes. Thus, the city area is divided into $2^\tau \times 2^\tau$ grids comprising of a grid system. The grid coordinates are numbered uniquely from 0 to $2^\tau - 1$.

In our proposed scheme, we construct a linear quadtree from the city area. Since each distinct poi is located in a unique leaf node of the linear quadtree, we can use a tuple of grid coordinates (x_i, y_i) to represent the location coordinates (lat_i, lon_i) of the poi in the city area. Thus, the GPS location information of poi can be represented in the form of grid location coordinates in our grid system.

C. MORTON CODE

G.M Morton [41] first proposed the concept of Morton code in 1966. Morton code is often used to map multidimensional data to one dimension while preserving the locality of the data point. It can be applied to generate a unique index for a tuple integer numbers. For example, the encoded value of a point in the two-dimensional space can be uniquely indexed. These indexes values are sorted in a “Z" shape. Fig. 3 illustrates the space partition and the corresponding decimal Morton code of the linear quadtree, where the depth of the linear quadtree is 3. The z-order of Morton codes have such an excellent characteristic that the coordinates of the adjacent Morton code numbers are also spatially close to each other in the multidimensional space. In recent years, the Morton code has been extensively used in computer graphics, such as tree construction, raster data compression, and spatial sorting.

In our proposed scheme, we utilize the Morton code's charming feature to find the nearest neighborhood poi record.The basic idea of the search nearly point is to test whether the Morton code value of a point is equal or close to another point in the grid system. The z-value of a point in two-dimensional space can be calculated by interleaving the binary representation of its coordinate values. We assume that II and JJ represent the row and column number of a point in the two-dimensional space, respectively. Given a row coordinate II of n bits whose binary presentation is $II = (i_1 i_2 \cdots i_n)$ and a column coordinate JJ of n bits whose binary presentation is $JJ = (j_1 j_2 \cdots j_n)$, the decimal Morton code is $M = (i_n j_n i_{n-1} j_{n-1} \cdots i_3 j_3 i_2 j_2 i_1 j_1)_2$. For example, if the gird coordinate of a point p_1 is $(3, 4)$ as shown in Fig.3, the binary representation of the row and column coordinate is 011 and 100, respectively. The Morton code value of this point is $M_1 = (011010)_2 = 26$.

V. OVERVIEW OF PPMQ SCHEME

In this section, we first introduce the location representation model. Then, we describe the formal definition of the pro-

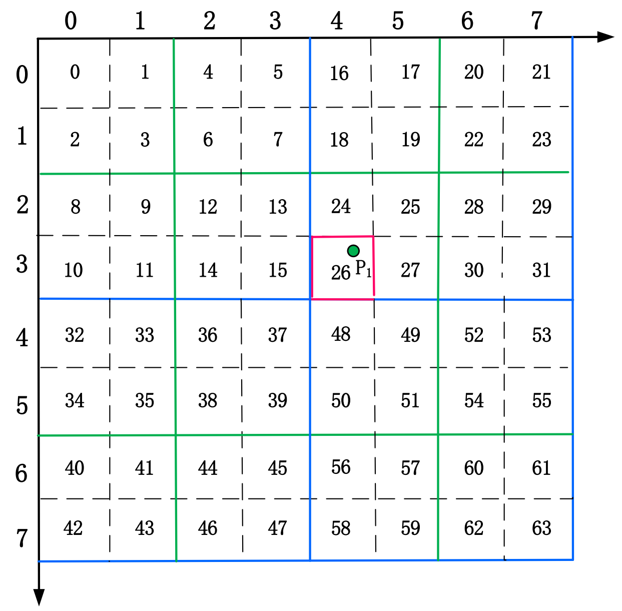


FIGURE 3. The decimal Morton code of the linear quadtree.

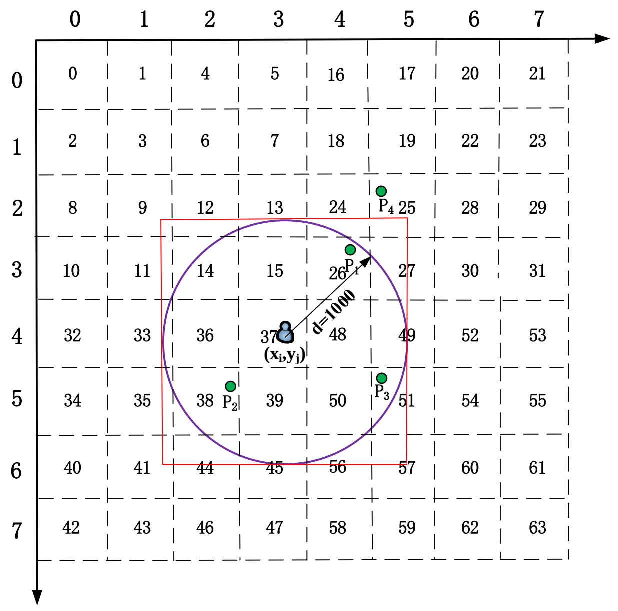


FIGURE 4. The details of the grids system.

posed scheme.

A. LOCATION REPRESENTATION MODEL

In this paper, we utilize the quadtree technique to describe the location coordinate of the poi in the searchable index $\mathcal{I}$ and encrypted query condition. For simplicity, we construct a balanced linear quadtree with depth τ based on the city area. That is to say, the city area has been recursively divided τ times. Thus, the city area is partitioned into $\mathbb{N} \times \mathbb{N}$ grids, where $\mathbb{N} = 2^\tau$, the length of the basic grid is denoted as σ. The grid coordinates are numbered uniquely from 0 to $\mathbb{N} - 1$. As described above, we use a tuple of coordinates to represent a point in the city area's gird system. The GPS location latitude and longitude coordinates of the poi can be represented as the grid coordinates of the poi. Fig.4 shows a grid system for the linear quadtree, where $\mathbb{N} = 8$. In Fig.4, the grid coordinates of the point p_1, p_2, p_3, p_4 can be expressed as $(3, 4)$, $(5, 2)$, $(5, 5)$, $(2, 5)$, respectively.

To improve the query process's efficiency, we encode the quadtree leaf nodes based on the Morton code. As we can see from Fig. 4, the Morton value of the point p_1, p_2, p_3, p_4 is 26, 38, 51 and 25 respectively. The two-point location coordinates, which are close to each other in the two-dimensional space, have Morton values that are close to each other. To achieve location query, the data user can specify his/her region of interest, which can be represented as a circle centered at the query user's current location $l_i = (x_i, y_i)$ with a radius of d in our grid system. The radius d can be regarded as a search range. Based on the query location $l_i = (x_i, y_i)$ and radius d, the data user can compute the minimum bounding rectangle, where the range of row and column number can be denoted as $[x_{min}, x_{max}]$ and $[y_{min}, y_{max}]$, respectively. Note that data users can adjust the size of the radius d. The larger the d, the more POIs will be searched. As a matter of simplicity, if most of a grid is covered by a data user's region of interest, this grid is added to the query region. For example, if the query location is $(4, 3)$ and $d = 1000$ meters, the region of interest of the data user is shown in Fig. 4. In Fig. 4, we can find that, the query range of the row and column for the data user is $[1, 5]$ and $[2, 6]$, respectively. After that, the set of Morton code values of query region is $QM = \{14, 15, 26, 36, 37, 38, 39, 45, 48, 49, 50\}$. The cloud server only needs to return the POIs, which meet the multi-keyword query condition, and the Morton code values of these POIs are in the set of Morton code values of query region to the data user.

As described above, the location of poi can be encoded as an integer number, and the query location range of the data users can be encoded as an integer set by the Morton encoding algorithm. We can covert the testing of whether the location of a point falls within a query region to the testing of whether the Morton code value of a point is one of the elements in the set of Morton code values of query region. That is to say, if a point is within the query region, the Morton code value of a point is a member of the set of Morton code values of the query region. A last, to protect location and query content privacy, we can design public key encryption to encrypt the set of Morton code values of query region and location coordinates of POIs.

B. FORMAL DEFINITION

The formal definition of the proposed scheme is defined as follows. Our scheme consists of six algorithms: **System Setup**, **User Register**, **Encryption**, **Generate Trapdoor**, **Query**, **User Revocation**.

- $System\ Setup(1^\lambda) \rightarrow (PK, MSK)$: The LBSP (i.e., data owner) takes a security parameter λ as input, and outputs public key PK, and master secret key MSK.

- $User\ Register(PK, MSK, u) \rightarrow (SK_u, ERK_u)$: The LBSP takes public key PK, master secret key MSK, a random number $s_u \in \mathbb{Z}_q^*$, a user-defined random number $r_u \in \mathbb{Z}_q^*$ as input, and outputs the user's ID UId_u, $g^{k\cdot r/s_u}$, and the user's search public key ERK_u. At last, the LBSP delivers UId_u, $g^{k\cdot r/s_u}$, hk_0, the grid system parameters τ, σ, and s_u to the user u via a secure communication channel and (UId_u, ERK_u) is sent to the cloud server by a secure channel.
- $Encryption(\mathcal{POI}, PK, MSK)) \rightarrow C = \mathcal{I}||\widetilde{\mathcal{POI}}$: Based on the LBS dataset $\mathcal{POI}$, the data owner first builds a searchable index $\mathcal{I}$. After that, the LBS dataset can be independently encrypted by a systematic encryption. Finally, the data owner obtains $\widetilde{\mathcal{POI}}$. At last, $C = \mathcal{I}||\widetilde{\mathcal{POI}}$ is outsourced to the cloud server.
- $Generate\ Trapdoor(\widehat{\mathcal{W}}, loc_i, d) \rightarrow T_{\widehat{W}}$: The data user first extracts t keywords from user's query content. With t keywords of interest in $\widehat{\mathcal{W}}$, the query location loc_i and query range d as input, it generates a corresponding trapdoor $T_{\widehat{W}}$.
- $Query(C, T_{\widehat{W}}) \rightarrow \widetilde{\mathcal{POI}}_{\widehat{W}}$: The cloud server takes the trapdoor $T_{\widehat{W}}$ and ciphertext C as input, and outputs the identifier list of relevant encrypted POIs, namely $\widetilde{\mathcal{POI}}_{\widehat{W}}$.
- $User\ Revocation(UId_u, UL) \rightarrow UL^{'}$: The cloud server takes the identify information UId_u of user u, the registered users list UL as inputs, and then delete corresponding user's identify information to obtain a new users list $UL^{'}$.

VI. CONSTRUCTION OF PPMQ SCHEME

In this section, we use the bilinear paring map and quad-tree technique to construct the PPMQ scheme. We present the construction of PPMQ scheme as follows:

A. SYSTEM SETUP

Given a system parameter λ, the LBSP first generates two multiplication cyclic group $\mathbb{G}$ and $\mathbb{G}_T$ with the large prime order q, and a bilinear paring map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where e is a non-degenerate bilinear pairing operation. Let g be a generator of $\mathbb{G}$. Then, the LBSP defines a random oracle $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$, a hash secret key hk_0, the grid system parameters σ, τ, and a key derivation function $KF(.)$, which are shared with the valid query users. Next, the LBSP generates a secret key $k \in \mathbb{Z}_q^*$, and a random number $r \in \mathbb{Z}_q^*$, which helps improve the flexibility and security of our system. Finally, the LBSP keeps the master key $MSK = (k, r)$ secretly, and opens the public key $PK = \{\mathbb{G}, \mathbb{G}_T, e, H_1, g, KF(.)\}$.

B. USER REGISTER

When a user u wants to enjoy the LBS, he/she first needs to register with the LBSP to obtain the search capability. The LBSP verifies the identity information of the user u. After the user u passed the authentication, the LBSP assigns a ID number UId_u to user u. Then, LBSP selects a random number $s_u \in \mathbb{Z}_q^*$ for u and compute $g^{k\cdot r/s_u}$. Next, the LBSP sends $g^{k\cdot r/s_u}$, s_u, hk_0, UId_u, and the grid system parameters τ, σ to the user u by a secure communication channel. When the user u receives $g^{k\cdot r/s_u}$, s_u, UId_u, hk_0, τ, and σ, he/she randomly selects $r_u \in \mathbb{Z}_q^*$ and then further computes g^{r_u} with keeping the private key $SK_u = \{r_u\}$ secretly. According to the received g^{kr/s_u}, he/she calculates his/her search public key ERK,

$$ERK_u = g^{k\cdot r/s_u} \times g^{r_u} = g^{k\cdot r/s_u + r_u} \tag{1}$$

At last, the user u also keeps (s_u, r_u, hk_0, σ, τ) secretly. (UId_u, ERK_u) is sent to the cloud server. The cloud server stores this tuple into a registered users list UL.

C. ENCRYPTION

To make the system secure and easy to search, before uploading POIs to the cloud server, the LBSP should first build a secure index $\mathcal{I}$ for encrypted data $\widetilde{\mathcal{POI}}$. As mentioned before, the searchable index $\mathcal{I}$ consists of the encrypted keywords $\widetilde{W}$ that describe the POIs in all aspect, and the encrypted Morton code value $\widetilde{M}$ of location coordinates of the POIs .

To enable the registered users to find the desirable POIs conveniently and quickly, the LBSP use several keywords to describe the POIs in a fine-grained model. Each keyword $w_{i,j} (1 \leq j \leq m)$ in W_i describes a certain aspect of the poi_i. For a point poi_i, the LBSP uses the following method to encrypt each keyword $w_{i,j} (1 \leq j \leq m)$ in W_i as follows.

$$\widetilde{w_{i,j}} = g^{k\cdot r\cdot H_1(w_{i,j})} \tag{2}$$

where $1 \leq j \leq m$, H_1 is a random oracle shared between LBSP and the registered data users, k is the secret key for the LBSP to encrypt keywords of POI, and $r \in \mathbb{Z}_q^*$ is a random number for the LBSP to improve the flexibility and security of our system.

To protect the location privacy, the LBSP first converts the GPS location coordinate of the poi_i to the grid coordinate based on the grid system. Then, the LBSP computes the Morton code value of the grid coordinate of the poi_i. At last, the LBSP uses the following method to encrypt the grid coordinate of poi_i.

$$\widetilde{M_i} = g^{k\cdot r\cdot H_1(Ed(x_i, y_j))} = g^{k\cdot r\cdot H_1(M_i)} \tag{3}$$

where $Ed(.)$ represents the Morton encoding algorithm, (x_i, y_j) is the grid coordinate of the poi_i, and M_i is the Morton code value of the grid coordinate of the poi_i. Through the above operations, the searchable index $\mathcal{I}_i$ has been built, which can be represented as $\mathcal{I}_i = (\widetilde{W}_i, \widetilde{M}_i)$.

To preserve the confidentiality of POIs, the LBSP encrypts each data item $poi_i \in \mathcal{POI}$ using the following formula.

$$\widetilde{poi}_i = Enc(poi_i, hk_i) \tag{4}$$

where $Enc(.)$ is a systematic encryption method, such as AES, DES, and hk_i is a secret key. The secret key hk_i can be obtained by a key derivation function $KF(.)$ shared between query user and LBSP. The secret key hk_i is generated as follows:

$$hk_i = KF(hk_0, M_i) \tag{5}$$

After that, we obtain the resultant ciphertext of encrypted POIs, denoted as $\mathcal{I}||\widetilde{\mathcal{POI}}$, where $||$ is concatenation character.

D. GENERATE TRAPDOOR

To implement the fine-grained query, the data user uses multiple keywords to describe the query requirements accurately. The data user first extracts t keywords from the user's query content, we call it query keywords. Then, the data user computes the query region based on the grid system, which is produced by the parameters σ and τ. To preserve the query content and location privacy, the data user encrypts each query keyword and query region before submitting a query request to the cloud. The data user takes two steps to generate a trapdoor. First, the data user encrypts the query keywords. Second, the data user uses the same method to encrypt the set of Morton code values of the query region.

In order to make the data users generate trapdoors securely, the query keywords encryption should satisfy two main conditions. First, for the same keyword, the data users can generate different trapdoor each time. Second, the data user does not need to ask the data owner for the secret key to generate a trapdoor. That is to say, the data user can generate a trapdoor independently. The data user encrypts each query keyword $w_i(1 \leq i \leq t) \in \widehat{W}$ as follows:

$$T_{w_i} = (g^{r^{'} \cdot s_u \cdot r_u \cdot H_1(w_i)}, g^{r^{'} \cdot s_u \cdot H_1(w_i)}, g^{r^{'}}) \tag{6}$$

where $r^{'} \in \mathbb{Z}_q^*$ is a random number and the value of $r^{'}$ is variable. The data user can set $r^{'}$ to a different value each time, which helps to improve the randomness and security of the query content.

To prevent the attackers from obtaining the location of the data user through analyzing the set of Morton code values of query region and disguising a valid query user to launch a query, the data user encrypts each Morton code value $M_j(1 \leq j \leq \mu) \in QM$ as follows:

$$\begin{aligned}\widetilde{M_j} &= (g^{r^{'} \cdot s_u \cdot r_u \cdot H_1(Ed(x_i, y_j))}, g^{r^{'} \cdot s_u \cdot H_1(Ed(x_i, y_j))}, g^{r^{'}}) \\ &= (g^{r^{'} \cdot s_u \cdot r_u \cdot H_1(M_j)}, g^{r^{'} \cdot s_u \cdot H_1(M_j)}, g^{r^{'}})\end{aligned} \tag{7}$$

where $Ed(.)$ represents the Morton encoding algorithm, (x_i, y_j) is the grid coordinate of the point in the query region, and M_j is the Morton code value of the grid coordinate of the point in the query region.

Through the above operations, the trapdoor has been generated, $T_{\widehat{W}} = (T_{w_1}, T_{w_2}, \cdots, T_{w_t})||(\widetilde{M_1}, \cdots, \widetilde{M_j}, \cdots, \widetilde{M_\mu})$

E. QUERY

After receiving a trapdoor $T_{\widehat{W}}$ from data user u, the cloud server searches $I_i \in \mathcal{I}$ one by one. The query process is conducted in three steps:

In the first step, the cloud server first reads the encrypted location dataset $\mathcal{I}||\widetilde{\mathcal{POI}}$. Next, the cloud server parses the ciphertext, and then gets the searchable index $\mathcal{I}$. Afterward, for each subindex I_i, the cloud server can easily obtain the $\widetilde{W}_i$ and $\widetilde{M}_i$ for the point poi_i.

In the second step, the cloud server tests whether the grid coordinate of the point poi_i is located in the query region of u. The cloud server will match the encrypted Morton code value $\widetilde{M}_i$ of the point poi_i with the element $\widetilde{M_j}$ in the set of encrypted Morton code values of query region $\widetilde{QM}$ using the following equation.

$$\begin{aligned}&e(g^{r^{'} \cdot s_u \cdot H_1(M_j)}, ERK_u) \\ &= e(g^{r^{'} \cdot s_u \cdot H_1(M_j)}, g^{k \cdot r/s_u + r_u}) \\ &= e(g^{r^{'} \cdot s_u \cdot H_1(M_j)}, g^{k\dot{r}/s_u}) \cdot e(g^{r^{'} \cdot s_u \cdot H_1(M_j)}, g^{r_u}) \\ &= e(g^{k \cdot r \cdot H_1(M_j)}, g^{r^{'}}) \cdot e(g^{r^{'} \cdot s_u \cdot r_u \cdot H_1(M_j)}, g) \\ &= e(g^{k \cdot r \cdot H_1(M_i)}, g^{r^{'}}) \cdot e(g^{r^{'} \cdot s_u \cdot r_u \cdot H_1(M_j)}, g) \\ &= e(g^{k \cdot r \cdot H_1(M_i)}, g^{r^{'}}) \cdot e(g^{r^{'} \cdot s_u \cdot r_u \cdot H_1(M_j)}, g)\end{aligned} \tag{8}$$

where ERK_u is the search public key of u. If the above equation holds, the location grid coordinate of the poi_i is located in the query region of u. The cloud server obtains all POIs that their location coordinates are located in the query region.

In the third step, the cloud server judges whether the keywords of the point match in the query keywords submitted by data users or not. The cloud server tests whether the keyword $w_{ji} \in W_j$ of the point poi_j is contained in the query keywords set $\widehat{W}$ or not. If the following equation holds, the keyword w_{ji} correctly matches the query keyword w_i.

$$\begin{aligned}&e(g^{r^{'} \cdot s_u \cdot H_1(w_i)}, ERK_u) \\ &= e(g^{r^{'} \cdot s_u \cdot H_1(w_i)}, g^{k \cdot r/s_u + r_u}) \\ &= e(g^{r^{'} \cdot s_u \cdot H_1(w_i)}, g^{k\dot{r}/s_u}) \cdot e(g^{r^{'} \cdot s_u \cdot H_1(w_i)}, g^{r_u}) \\ &= e(g^{k \cdot r \cdot H_1(w_i)}, g^{r^{'}}) \cdot e(g^{r^{'} \cdot s_u \cdot r_u \cdot H_1(w_i)}, g) \\ &= e(g^{k \cdot r \cdot H_1(w_{ji})}, g^{r^{'}}) \cdot e(g^{r^{'} \cdot s_u \cdot r_u \cdot H_1(w_i)}, g)\end{aligned} \tag{9}$$

Note that, a point poi_j cloud be returned if and only if it has at least one keyword matching the user's query keywords. If the above equation holds, w_i is equal to w_{ji}. After that, the cloud server filters unqualified results obtained in the previous step. With these three steps, the cloud server has found all qualified POIs that their location and keywords match the query condition, and then returned the qualified encrypted POIs to the querying user u. The query user uses hk_0 and corresponding Morton code value M_i of the poi_i to generates the decrypt secret key hk_i. Lastly, the querying user utilizes hk_i to recover the plaintext of the poi_i by the

AES decryption algorithm. The could server cannot obtain any sensitive data from the returned POIs.

F. USER REVOCATION

User revocation is an important and challenging take in an LBS system. If the LBSP wants to revoke a user u, the LBSP first sends the ID of user u, UId_u, to the cloud server. Then, the cloud server scans users' information in the registered users list UL to find out the information of user u. Next, the cloud server deletes (UId_u, ERK_u) to obtain a new users list, denoted as $UL^{'}$. Once (UId_u, ERK_u) is deleted from users list stored at the cloud server, the data user u no longer has the search capabilities to query the encrypted LBS location data. Since, without ERK_u, the cloud server cannot perform keywords matching between trapdoor and encrypted query keywords. Once the LBSP has revoked u, he/she can still generate a legal trapdoor. However, he/she no longer has the capability to search the encrypted POIs. The cloud server can reject the query request from the data user u.

VII. SECURITY ANALYSIS

In this section, we step by step analyze our proposed scheme's security to demonstrate that the security and privacy requirements have been satisfied for the POIs, the keywords, the queries, and the location information.

POIs : In our proposed scheme, these POIs are encrypted by the semantically secure symmetric encryption algorithm, such as AES, DES, before uploaded to the cloud server. As long as the encryption algorithm is secure, the attacker cannot know the actual content of poi. Since the secret key hk_i is keeping secretly by the data user who has been registered to the LBSP, the unauthorized data user is hard to obtain the actual contents of these POIs.Thus, POIs are protected from unauthorized access. The privacy of POIs is preserved.

Keywords : The keywords that describe the poi record from all aspects are encrypted before uploaded to the cloud server. Let us consider a popular game played between a challenger $\mathcal{C}$ and a probabilistic polynomial-time (PPT) attacker $\mathcal{A}$. In our scheme, the cloud server can be regarded as an attacker $\mathcal{A}$. The LBSP or authorized data user can be acted as the challenger $\mathcal{C}$. For keywords encryption, $\mathcal{C}$ first generates the following parameters, k, r, g, q, and a random oracle H_1, Then, $\mathcal{C}$ makes these parameters public. Next, $\mathcal{C}$ selects a keyword $w_{ij}^{'}$ from W_i of the poi_i, and subsequently sends $g^{k \cdot r \cdot H_1(w_{ij}^{'})}$ to $\mathcal{A}$. Based on this information, $\mathcal{A}$ would try to guess $w_{ij}^{'}$. However, the Decisional Diffie-Hellman Problem (DDHP) is hard; it is difficult to obtain $g^{H_1(w_{ij}^{'})}$. As the Discrete Logarithm Problem (DLP) is also hard, it is intractable to compute $H_1(w_{ij}^{'})$ in polynomial time. Even if the ciphertext of the keywords has been stripped, the attacker obtains the hash value of keywords, $H_1(w_{ij}^{'})$. Due to the one-wayness and collision resistance properties of the hash function, $\mathcal{A}$ cannot recover the keyword $w_{ij}^{'}$ by semantic analysis. Our scheme implements semantic security against adaptive chosen keyword attack (IND-CKA) secure using random oracle and bilinear paring technique. Therefore, the security of the keywords is well preserved.

Trapdoors : The security of trapdoor can be analyzed from three aspects. Recall the trapdoor construction formula,

$$T_{w_i} = (g^{r^{'} \cdot s_u \cdot r_u \cdot H_1(w_i)}, g^{r^{'} \cdot s_u \cdot H_1(w_i)}, g^{r^{'}}) \tag{10}$$

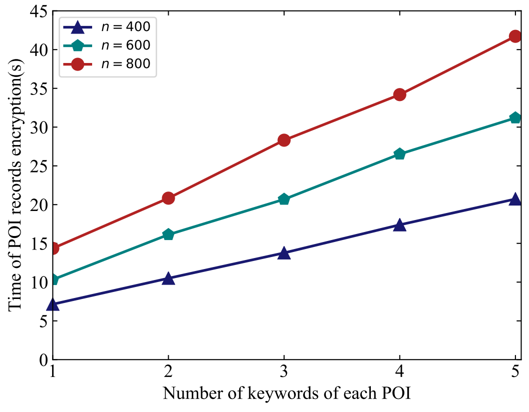
On the one hand, as the discrete logarithm problem is hard to solve, the attacker is difficult to obtain $H_1(w_i)$. The attacker must have known something about $H_1(.)$ and using semantic analysis to guess the queried keyword w_i. However, the attacker has no idea about the $H_1(.)$, it is infeasible to recover w_i. On the other hand, in our scheme, only the registered data users and LBSP knows the random oracle $H_1(.)$. Without knowing $H_1(.)$, the attacker cannot generate a correct trapdoor for the chosen keyword. It is inapplicable to try out many possible keyword values to find out w_i. Besides, the attacker would try to infer sensitive information based on query results. The attacker would go to a specific place to obtain the corresponding POIs and then guess the query content of the data users. Nevertheless, this attack is impractical. The reason is that, to prevent the attacker from knowing the query content, we insert a random number $r^{'}$ to encrypt the keyword, which can blur the encrypted keywords. The same keyword can be encrypted into different ciphertext each time. For the encrypted location information $\widetilde{QM}_j$, we can adopt the same method to analyze the security of $\widetilde{QM}_j$. Relying on the semantic analysis, the attacker cannot distinguish a specific encrypted query keyword from the trapdoors, the attacker cannot guess the query content of the data user. Hence, the query privacy is preserved.

Location Information: In our scheme, the location information is encoded with the Morton coding algorithm and then encrypted in the same way as the keyword encryption. Since the DDHP is hard, the location information encryption algorithm is IND-CKA secure. The reason is demonstrated as follows. We consider a game played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, which acts as the authorized data user and LBSP in the location information encryption algorithm. We assume that adversary $\mathcal{A}$ has a non-negligible advantage ϵ as the attacker in this game. The game is conducted in the following steps.

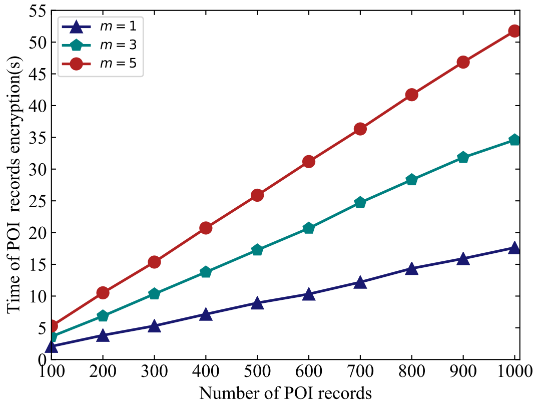
Step 1: $\mathcal{C}$ runs the system setup algorithm to generate the public parameters $(g,\ q,\ H_1,\ e,\ \mathbb{G},\ \mathbb{G}_T, g^r, C_1)$, and then sends these parameters to $\mathcal{A}$.

Step 2: $\mathcal{A}$ generates two location coordinates, $l_0 = (x_0, y_0), l_1 = (x_1, y_1)$ and then sends l_0, l_1 to $\mathcal{C}$. Next, $\mathcal{C}$ randomly chooses $l_b (b \in 0, 1)$ and encodes it into $M_j (j = 0, 1)$. Finally, $\mathcal{C}$ encrypts M_j by Eq. 3 and gives $\widetilde{M_j} = g^{k \cdot r \cdot H_1(M_j)}$ to $\mathcal{A}$.

Step 3: Based on these information, $\mathcal{A}$ would try to calculate M_j and outputs the guess $j^{'} \in \{0, 1\}$ for j. We denote t as the bit that $\mathcal{A}$ is trying to guess. When $t = 0$, C_1 is given $g^{k \cdot r}$; Otherwise C_1 is set randomly in $\mathbb{G}$. The adversary outputs 0 if and only if $j^{'} = j$. If $j^{'} = j$, we can say the adversary $\mathcal{A}$ wins the game. Let $Pr[j^{'} = j]$ be the probability



(a) For different number of keywords of each POI



(b) For different number of POI records

FIGURE 5. Time cost of POI encryption

that the adversary guesses correct. The advantage of $\mathcal{A}$ that wins this game is $Adv_{\mathbb{A}} = |Pr[j^{'} = j] - 1/2|$.

Proof: If $t = 0 (i.e., C_1 = g^{k \cdot r})$, it means that $\widetilde{M_j} = C_1 \cdot g^{H_1(M_j)}$ is a valid location information encryption. In this sense, the adversary $\mathcal{A}$ guess correctly with probability $1/2 + \epsilon$. If $t = 1$, the adversary receives the ciphertext $C_1 \cdot g^{H_1(M_j)} = g^{r^{'}}$, where $r^{'}$ is a random number in $\mathbb{Z}_q^*$. In this case, the value of j is hidden to $\mathcal{A}$, $\mathcal{A}$ will guess it correctly with probability 1/2. Hence, $Pr[j^{'} = j] = 1/2 \cdot (1/2 + \epsilon) + 1/2 \cdot 1/2 = 1/2 + \epsilon$.

Since ϵ is non-negligible, $Adv_{\mathbb{A}} = |Pr[j^{'} = j] - 1/2| = \epsilon$. The advantage of $\mathcal{A}$ to win this game is non-negligible. This conclusion violates the assumption that DDHP is hard. Thus, our location information encryption is semantically secure. The privacy of the location is preserved.

VIII. PERFORMANCE EVALUATIONS

In this section, we measure the efficiency of the PPMQ in terms of the encryption of POIs time, trapdoor generation time, and query processing time with a real LBS dataset. The proposed scheme supports multi-user settings and provides a flexible multi-keyword location query service to data users.

A. SIMULATION EXPERIMENT SETTINGS

We conduct a thorough performance experimental evaluation of the proposed scheme on a real LBS data set, the OpenStreetMap project in Singapore [42]. The dataset's POIs are extracted from the LBS resource items in Singapore, which has 32730 POIs. Most POIs in the dataset have less than 5 keywords, while a few of them may contain more than 5 keywords. In our experiment, we randomly register 100 users into the geographic area of Singapore. We build a linear quadtree and then obtain a grid system based on the geographic area of Singapore. We assume that the length of the basic grid is $\sigma = 1000$ meters. Then, the number of recursive divisions of the geographic area can be set to $\tau = 5$. For every 5 minute, 10% of users are randomly selected to issue queries.

The experiment programs are coded using JAVA programming language on a PC running JDK1.8 platform with 3.30GHz Intel(R) Xeon(R) E3-1225 CPU, 8G memory, and a Linux Mint 17.3 Rosa operation system. We use the type-A elliptic cure parameter, where the group order q is 160-bits, and SHA1 as the random oracle for hashing keywords and Morton code value. We use the AES algorithm to encrypt POIs records with a 128-bit key. In our evaluations, we implement the pow and paring operation under type-A parameters without preprocessing. All the experiments are run 10 times to calculate the average time cost in different phases. We implement all necessary routines for LBS providers to process location data such as encryption of POIs, data users to generate trap doors, and the cloud server to perform the multi-keyword searches.

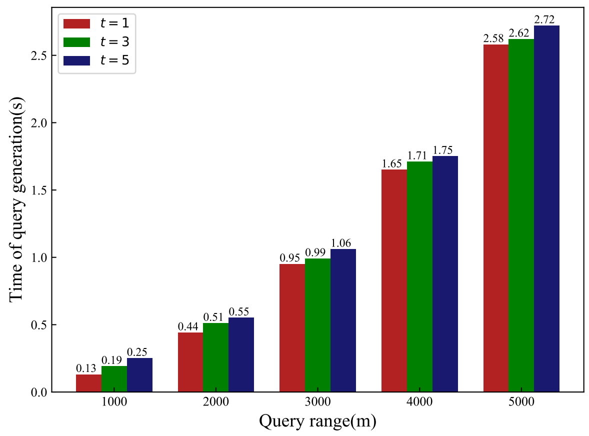
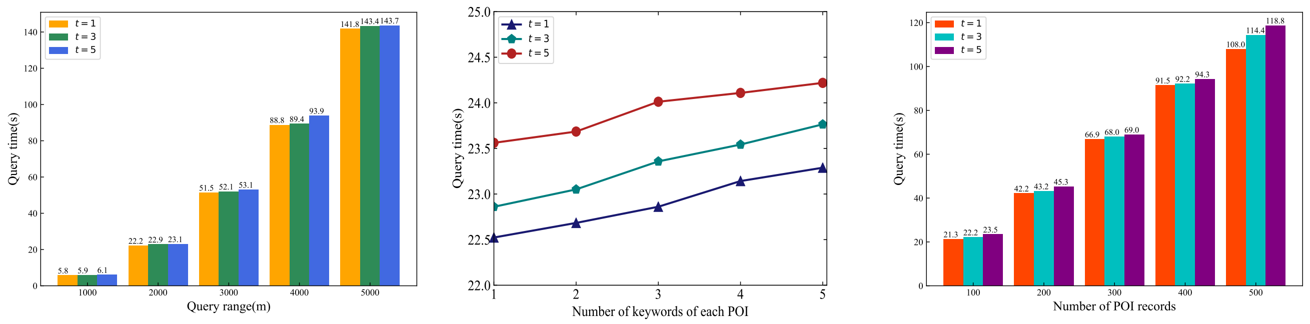


FIGURE 6. Time cost of query generation.



(a) For different query range with fixed number of POI records and the same number of POI keywords, n=100, m=5

(b) For different number keywords of each POI with fixed number of POI records and the same query range, n=100, d=2000

(c) For different number of POI records with fixed query range and the same number of POI keywords, d=2000, m=5

FIGURE 7. Time cost of query. Note that the time cost should much lower once deployed at a real cloud

B. THE TIME COST OF DATA PROVIDER

In our proposed PPMQ scheme, the main operations of the LBSP is to prepare POIs. Before outsourced these POI records to the cloud server, we should encrypt these POI records. The encryption execution at the LBSP consists of building secure index for these POI records and encryption of them. The factors affecting the computation cost of encryption algorithm are the number of POIs n, the number of keywords m of each POI record. Thus, we test the efficiency with the different number of keywords m of each POI, and different number of POI records n, respectively. The number of keywords of each POI is selected from 1 to 5. As shown in Fig. 5(a), the time costs of POI records encryption increase with m. Fig. 5(b) shows that, with the n increases, the time cost of POI records encryption increases linearly approximately. We can see that, given $m = 5$, when n increases from 100 to 1000, the average time costs increases from 5.27s to 51.75s. The reason is that a larger m or n results in a longer POI records ciphertext. Hence, more time cost is needed to construct the secure searchable index and encrypt the POI records.

C. THE TIME COST OF DATA USER

The primary operations of the data user are to generate the trapdoor. To see whether the time cost is acceptable for mobile LBS user or not, we measured the time cost of trapdoor generation. The different number of query keywords t and query range d are chosen to illustrate the time cost of the data user. To observe the query generation's time cost, five query ranges are uniformly selected from 1000 to 5000 meters, and then 10 different query coordinates are randomly selected from the open street map of Singapore for each query range. The number of query keywords increases from 1 to 5. Next, for different query keywords and query range, we execute 10 times trapdoor generation algorithm with different query coordinates and calculate the average time cost for each query condition. Fig.6 shows the time cost of generating query conditions. From Fig. 6, we can observe that the time of generating trapdoor increases as t increases. We can also observe that from Fig. 6, with the query range d increases, the time cost of generating trapdoor also increases. That is because the larger the number of query keywords t, the more encrypt operation computations are required. Given a fixed t, the larger the query range d, the more time cost is needed to construct the query region and encrypted the Morton values of the query region. When the query range d increases from 1000 to 5000 meters, the average trapdoor generation time cost is less than 2.72s. In general, the data user will often choose very seldom query keywords and a relatively small query range to search the desirable POIs. Note that, when the query range $d <= 2000$, the time cost of query generation is about 0.5s. Thus, the time cost in data users is acceptable for mobile devices.

D. THE TIME COST OF CLOUD SERVER

We now consider the query algorithm. The query algorithm execution at the cloud server consists of matching location and keywords. The cloud server computes a bilinear paring over a prime order group for each location coordinate in the query region to match the location information. Then, the cloud server matches the query keywords with the keywords of each POI. The number of query keywords t increases, the number of keywords m of each POI, query range d, and the number of POI records n may impact the computation complexity in the cloud server. Therefore, different t, m, d and n are choose to illustrate the time cost of query. Both t and m varies from 1 to 5. The number of POI records n grows from 100 to 500. Form Fig. 7(a), we can see that, given $m = 5$ and $n = 100$, the time cost increases as the number of query keywords t increases and when the query range d increases, the time cost of query also increases. The influence of the query range parameter is greater than the number of query keywords. That is, because the larger the query range d, the more POIs need to be matched, the more query operations, the cloud server needs to be performed. Fig. 7(b) illustrates that given $n = 100$ and

$d = 2000$, the time cost in cloud servers rises with the increasing in the number of query keywords t. When the number of keywords of each POI m increases, the time cost increases linearly approximately. Since the larger the m, the more keywords matching operations the cloud server needs to perform. Fig. 7(c) demonstrates that, given a fixed $d = 2000$ and $m = 5$, with the number of POI records increases, the time cost also increases. Furthermore, the more number of query keywords, the more time cost is required for keyword matching. The reason is that the larger the n, the more time cost is needed to match the query keywords and location information. Note that our experiments were simulated on a PC, which plays the role of cloud, and only one CPU core can be utilized to computing. Our scheme's performance will be perfect on a real cloud server, which has much more computing resources.

IX. CONCLUSION

In this paper, we explore the problem of the multi-keyword query in LBS. Different from prior works, we have proposed a novel efficient and privacy-preserving multi-keyword query scheme in LBS over the outsourced cloud, named PPMQ, which preserves location and query content privacy, and achieves confidentiality of location data. The designed scheme can prevent the LBS provider or unregistered users from deducing the query content. The authorized data user can obtain accurate LBS query results without divulging his/her location information efficiently. Specifically, we developed a flexible user registration and a user revocation mechanism, the proposed scheme is scalable. Furthermore, we give security analysis and conduct extensive experiments on a real LBS data set to evaluate our scheme's performance, and experimental results demonstrate the efficacy and efficiency of our proposed scheme. In the future work, we will take into consideration of the integrity verification for the query results.

REFERENCES

[1] Statista Reaserch Department, "Number of location based service users in the United States from 2013 to 2018," Statista, 2015, Available: https://www.statista.com/statistics/436071/location-based-service-users-usa.

[2] L. Li, R. Lu and C. Huang, "EPLQ: Efficient Privacy-Preserving Location-Based Query Over Outsourced Encrypted Data," in IEEE Internet of Things Journal, vol. 3, no. 2, pp. 206-218, April 2016.

[3] W. Liang, Y. Fan, C. Li,D. Zhang, J.-L.Gaudiot. "Secure Data Storage and Recovery in Industrial Blockchain Network Environments," IEEE Transactions on Industrial Informatics. 2020, pp.1-10, doi:10.1109/ TII. 2020. 2966069.

[4] J. Shao, R. Lu, and X. Lin, "FINE: A fine-grained privacy-preserving location-based service framework for mobile devices," in Proc. IEEE INFOCOM, pp. 244–252, 2014.

[5] W. Liang, K.C. Li, J. Long, X. Kui, Zomaya. "An Industrial Network Intrusion Detection Algorithm based on Multi-Characteristic Data Clustering Optimization Model," IEEE Transactions on Industrial Informatics, 2019.10, doi:10.1109/TII.2019.2946791.

[6] S. Zhang, Y.Lin, Q. Liu et al. "Secure hitch in location based social networks," Computer Communications, vol.100, pp.65-77, 2017

[7] B. Liu, W. Zhou, T. Zhu, L. Gao and Y. Xiang, "Location Privacy and Its Applications: A Systematic Study," in IEEE Access, vol. 6, pp. 17606-17624, 2018.

[8] W. Liang, W. Huang, J. Long, K.C Li, D. Zhang. "Deep Reinforcement Learning for Resource Protection and Real-time Detection in IOT Environment," IEEE Internet of Things Journal, 2020.1, doi:10.1109/JIOT.2020. 2974281

[9] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," IEEE Trans. Mobile Comput., vol. 7, no. 1, pp. 1–18, Jan. 2008.

[10] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," ACM Trans. Knowl. Discovery Data, vol. 1, no. 1, p. 24-26, Mar. 2007.

[11] N. Li, T. Li and S. Venkatasubramanian, "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity," 2007 IEEE 23rd International Conference on Data Engineering, Istanbul, 2007, pp. 106-115.

[12] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in Proc. IEEE ICPS, Jul. 2005, pp. 88–97.

[13] C. A. Ardagna, M. Cremonini, S. D. C. di Vimercati, and P. Samarati, "An obfuscation-based approach for protecting location privacy," IEEE Trans. Depend. Sec. Comput., vol. 8, no. 1, pp. 13–27, Jan./Feb. 2011.

[14] A. Pingley, N. Zhang, X. Fu, H.-A. Choi, S. Subramaniam, W. Zhao, "Protection of query privacy for continuous location based services," in INFOCOM, 2011 Proceedings IEEE. IEEE, 2011, pp. 1710–1718

[15] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in Proc. 14th Annu. ACM Int. Symp. Adv. Geograph. Inf. Syst., Arlington, VA, USA, 2006, pp. 171–178

[16] H.To, G.Ghinita, L.Fan, C.Shahabi,"Differentially private location protection for worker datasets in spatial crowdsourcing," IEEE Trans. Mobile Comput., vol. 16, no. 4, pp. 934–949, Apr. 2017.

[17] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location based systems," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2013, pp. 901–914.

[18] M. R. Paulet, G. Kaosar, X. Yi, and E. Bertino, "Privacy-preserving and content-protecting location

based queries," IEEE Trans. Knowl. Data Eng., vol. 26, no. 5, pp. 1200–1210, May 2014.

[19] G.Ghinita, P.Kalnis, A.Khoshgozaran, C.Shahabi, K.-L.Tan, " Private queries in location based services: Anonymizers are not necessary," in Proc. ACM SIGMOD, 2008, pp. 121–132.

[20] S Zhang, X. Li, Y. Lin, et al. "A privacy-preserving friend recommendation scheme in online social networks." Sustainable cities and society 38 (2018): 275-285.

[21] H. Zhu, R. Lu, C. Huang, L. Chen, and H. Li, "An efficient privacy-preserving location-based services query scheme in outsourced cloud," IEEE Trans. Veh. Technol., vol. 65, no. 9, pp. 7729–7739, Sep. 2016.

[22] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Enhancing privacy through caching in location-based services," in Proc. IEEE INFOCOM, Apr./May 2015, pp. 1017–1025.

[23] B. Liu, W. Zhou, T. Zhu, L. Gao, T. H. Luan, and H. Zhou, "Silence is golden: Enhancing privacy of location-based services by content broad- casting and active caching in wireless vehicular networks," IEEE Trans. Veh. Technol., vol. 65, no. 12, pp. 9942–9953, Dec. 2016.

[24] B. Liu, W. Zhou, T. Zhu, H. Zhou, and X. Lin, "Invisible hand: A privacy preserving mobile crowd sensing framework based on economic models," IEEE Trans. Veh. Technol., vol. 66, no. 5, pp. 4410–4423, May 2017.

[25] C. Wang, N. Cao, J. Li, K. Ren and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," 2010 IEEE 30th International Conference on Distributed Computing Systems, Genova, 2010, pp. 253-262.

[26] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in Proc. of ACM CCS'06, 2006, 79-88

[27] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," IEEE Transactions on Parallel and Distributed Systems, 2014, vol. 25, no. 1, pp. 222–233.

[28] R. L. K. Y. Z. Xu, W. Kang and C. Xu, "Efficient multi-keyword ranked query on encrypted data in the cloud," in Proc. IEEE Parallel and Distributed Systems (ICPADS'12), Singapore, Dec. 2012, pp. 244–251

[29] W. Zhang, Y. Lin, S. Xiao, J. Wu and S. Zhou, "Privacy Preserving Ranked Multi-Keyword Search for Multiple Data Owners in Cloud Computing," in IEEE Transactions on Computers, 2016, vol. 65, no. 5, pp. 1566-1577

[30] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in Proc. IEEE INFOCOM'10, San Diego, CA, Mar. 2010, pp. 1–5.

[31] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," in Proc. IEEE 31th Inter- national Conference on Distributed Computing Systems (ICDCS'11), Minneapolis, MN, Jun. 2011, pp. 383–392.

[32] P. Golle, J. Staddon, and B. R. Waters, "Secure Conjunctive Keyword Search over Encrypted Data," in Proc. of ACNS'04, 2004, pp. 31–45.

[33] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in Proc. of ICICS'05, 2005

[34] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in Proc. of EUROCRYPT, 2010.

[35] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in Proc. of TCC, 2009.

[36] V. K.A Sandor, Y Lin, X Li , et al. "Efficient decentralized multi-authority attribute based encryption for mobile cloud data storage". Journal of network and computer applications, 2019, 129:25-36.

[37] A. O. Freier, P. Karlton, and P. C. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0," RFC 6101 (Historic), Internet Engineering Task Force, Aug. 2011.

[38] T. Ylonen, "The Secure Shell (SSH) Protocol Architecture," IETF RFC:4251, Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4251.txt

[39] J. Hur and D. K. Noh. "Attribute-based access control with efficient revocation in data outsourcing systems," IEEE Trans. Parallel Distrib. Syst., 2011, 22(7):1214-1221

[40] I. Gargantini, "An effective way to represent quadtrees", ACM Commun. vol. 25, no. 12, pp. 905–910, 1982.

[41] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," IBM Ltd, Ottawa, Canada, Tech. Rep., 1966.

[42] Openstreetmap Foundation, West Midlands, U.K., Openstreetmap, 2020. [Online]. Available: http://www.openstreetmap.org