Fine-Grained Query Authorization With Integrity Verification Over Encrypted Spatial Data in Cloud Storage

Feng Tian, Zhenqiang Wu, Xiaolin Gui, Jianbing Ni, *Member, IEEE*, and Xuemin (Sherman) Shen, *Fellow, IEEE*

Abstract—In this article, a fine-grained query authorization scheme with integrity verification is proposed over encrypted spatial data for location-based services (LBS). The fine-grained query authorization is enabled based on a distribution of the spatial data by employing a non-uniform partition in the spatial domain to generate a density-based space filling curve (DSC), which can be used to generate index values for querying and transformation keys. The transformation keys can be used to generate query tokens for a secure spatial query as well as construct a transformation key tree whose subtree can be distributed by the LBS provider to an authorized user as transformation key for query tokens generation. Furthermore, the proposed scheme constructs a Merkle quad tree (MQ-tree) to support integrity verification by aggregating a digest of the spatial data based on the DSC and employing the MQ-tree as a verification structure. The LBS provider can share a subtree of the MQ-tree to authorized user as his verification structure, which corresponds to the transformation key of the authorized user. In this way, the authorized user can only generate the valid query tokens and verify the query results in his authorized region. The security properties of the proposed scheme is discussed, and extensive experimental results demonstrate the high efficiency of verification structure generation and verification operations.

Index Terms—Cloud storage, spatial data, integrity verification, location-based services, space filling curve

1 Introduction

A tremendous amount of spatial data is accumulated via smartphones and exploited to offer various location-based services (LBS), e.g., point of interests (POI) discovery, proximity-based notification, and location-based mobile advertising [1], [2]. Some of these services are provided to third-party companies for profits. For instance, Foursquare provides a global places database (containing 105 million POIs, available across 170 countries and territories) to 100,000 companies and application developers [3] that can analyze visit and venue data to identify shopper and consumer insights trends. As the volume of spatial data is growing rapidly, the LBS provider, denoted as data owner, is unable to support the storage of the tremendous volume of data. Thus, outsourcing the spatial data to the cloud is becoming a prevailing paradigm.

The outsourced data is encrypted by the data owner before it is outsourced to the cloud in order to protect its confidentiality. Moreover, indexes of the encrypted spatial data are also built by the data owner to enable its querying in encryption mode. Moreover, for flexible data sharing, the data owners desire fine-grained query authorization that only allows their authorized users to search their specific data [4], [5]. As the data is queried by users with different levels of trust, query authorization should be fine-grained to offer different users with different query capabilities, thereby restricting the users from querying the data outside their authorized regions. However, it is challenging to provide fine-grained query authorization over the encrypted spatial data since the distance relationship of the spatial data should be preserved in the encryption mode while considering the data distribution in the sub regions.

Searchable Encryption (SE) [6] supports the query capabilities over the encrypted data at the cloud without decryption. Nevertheless, most of the SE schemes focus on SQL queries, and cannot be directly employed to spatial data because of the completely different relationship among the data. To enable query services on encrypted spatial data, space filling curves have been widely used to transform the original locations of POIs to one-dimensional index values. Space filling curve passes through every partition of a closed space, and has no intersection with itself. In this way, each point in multi-dimensional space will be mapped as a value to one-dimensional space. Standard Hilbert curve (SHC) as a form of space filling curve is applied as a building block in

- *Feng Tian and Zhenqiang Wu are with the School of Computer Science, Shaanxi Normal University, Xi'an, Shaanxi 710062, China. E-mail: {tianfeng, zqiangwu}@snnu.edu.cn.*
- *Xiaolin Gui is with the School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China. E-mail: xlgui@mail.xjtu.edu.cn.*
- *Jianbing Ni is with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada. E-mail: jianbing.ni@queensu.ca.*
- *Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: xshen@bbcr.uwaterloo.ca.*

Manuscript received 28 Mar. 2019; revised 26 June 2020; accepted 14 July 2020. Date of publication 21 July 2020; date of current version 6 Sept. 2022. (Corresponding author: Zhenqiang Wu.) Recommended for acceptance by D. Thain. Digital Object Identifier no. 10.1109/TCC.2020.3010915

many schemes [1], [7], [8], [9], [10], [11] for spatial data transformation, which can protect the confidentiality of outsourced spatial data and make effective spatial queries. With the transformation key and the original spatial query, users can generate the query tokens to search over the encrypted spatial data. These query tokens can be used by the cloud to make equality checking over the query indexes of the encrypted data and confirm that the data belong to the query result. However, the SHC-based query tokens are generated under a single transformation key. As a result, by distributing the transformation key, the data owner can only authorize the users to query in the entire spatial domain, instead of the sub regions specified by the data owner. This query capability authorization is defined as coarse-grained. If the data owner can determine the regions that the users can generate the query tokens, then this query capability authorization is defined as fine-grained.

In addition, as authorized users utilize query services to obtain their interested data, the received query results should contain all spatial data points satisfying the query condition without any tampering. However, for economic benefits, the cloud may modify the query results for the paid advertiser (competitor that promotes its wanted POIs through extra fees) [9], and this kind of behavior is called data tempering attacks [12]. For example, it may inject extra data points or delete the top-ranking data points, which benefits the paid advertiser and decreases the utility of the query results. Therefore, the integrity verification of the query results is very important in the spatial data outsourcing. Moreover, in consistent with the fine-grained query authorization, the verification capabilities of users should also be fine-grained to guarantee that the users can only verify the query results in their authorized regions. Nevertheless, it is not trivial to design a fine-grained integrity verification approach over the encrypted spatial data. Since the verification structure for spatial data should contain the relationship among the spatial data, which is different from the relationship among the string or numeric values, the conventional integrity verification schemes [13], [14], [15], [16], [17], [18], [19], [20] cannot be directly applied to verify the spatial query results. SHC-based probabilistically replicating scheme (SPR) [9] achieves query authorization and integrity verification over encrypted spatial data. However, as a type of SHC-based scheme, SPR cannot support fine-grained verification capability authorization, which is challenging for existing SHC-based schemes due to their single transformation key. In addition, SPR is not a deterministic integrity verification scheme, because it introduces false negative (i.e., attackers may escape from being detected) among the results verification.

In this paper, a fine-grained query authorization scheme with integrity verification is proposed based on density-based space filling curve (DSC)[21]. The proposed scheme extracts the distribution of the outsourced POIs, and employs the non-uniform partition in the spatial domain. Each partitioned region contains curve parameters for spatial transformation in that region, and the partitioned regions with the same size scale are merged. Meanwhile, the curve parameters of these merged regions are updated based on the parameters of their sub regions. These curve parameters are denoted as the transformation keys corresponding to these merged regions. The transformation keys are organized in a quadtree structure, and tree nodes at different levels correspond to various spatial regions, which enables the fine-grained query capability authorization. Moreover, the proposed scheme generates digests information for each partitioned region, which are then organized in a quadtree structure for query result verification. Tree nodes at different levels correspond to different spatial regions, and the corresponding digests are used for query result verification. Thus, the fine-grained verification capability authorization is supported, which means only the users with the verification structure corresponding to the authorized region can verify the integrity of the query result. The proposed scheme is suitable for the application where the LBS provider (data owner), such as Foursquare, provides POI data to the third-party companies and developers. In this application, the data owner outsources his spatial data periodically (e.g., weekly or monthly), which can be considered as a series of static datasets on which different transformation parameters and symmetric keys are employed by the data owner to obtain a series of transformation key trees, verification structures, and symmetric keys. The generated transformation keys, verification structures, and symmetric keys are then distributed to different authorized users to achieve fine-grained query and verification capabilities authorization. The contributions of this paper can be summarized as three folds.

- A spatial query processing approach is proposed based on DSC, which supports query token generation for a single POI, range query and KNN query processing. Each node in the transformation key tree corresponds to a specific spatial region. By checking the intersection regions between the authorized region and transformation key tree, this approach can determine the subtree that corresponds to the authorized region. Through the subtree distribution, this approach enables fine-grained query capability authorization in the spatial domain, which supports the data owner to offer different query capabilities to authorized users with different trust levels.
- A spatial query integrity verification approach is proposed, which includes the verification structure generation, the range, and KNN query verification algorithms. By extending the Merkle hash tree (MHT) [22], the proposed approach generates a Merkle quad tree (MQ-tree) as the verification structure, which aggregates the digests of the data points based on DSC to achieve fine-grained verification capability authorization. The authorized users can only verify the integrity of the query results in their authorized regions. Moreover, compared with the probabilistically replicating scheme, the verification approach does not introduce false negative, which means that any dishonest behaviors can be detected.
- The security properties of the proposed scheme is analyzed, and its performance is evaluated through extensive experiments and performance comparisons. The experimental results demonstrate that both the query index and verification structure generation time of the proposed scheme are lower than

that of SHC-based approach. Meanwhile, either the verification time or the storage cost of the proposed scheme is lower than that of SPR.

The remainder of this paper is organized as follows. The related work is reviewed in Section 2 and system model, threat model and design goals are formalized in Section 3. In Section 4, the fine-grained query authorization scheme with integrity verification is proposed and its security is discussed in Section 5, followed by the computation and storage performance evaluation on the index generation and integrity verification in Section 6. Finally, the conclusion is drawn in Section 7.

2 Related Work

In this section, the references are reviewed, including the query processing of outsourced data and the integrity verification of the query result.

2.1 Query Processing of Outsourced Data

Privacy protection of outsourced data is vital and has attracted a lot of attention from the academia and industry. Hacigümüs *et al.* introduced the idea of outsourcing database services to a third-party service provider (SP) [23] and provided an outsourced data privacy protection scheme [24]. This scheme supports encrypted data query by constructing index based on encrypted data and additional bucketing information. Later, aiming at numeric values, Agrawal *et al.* [25] proposed an order-preserving encryption scheme (OPES), which supports efficient processing of queries at the SP. To support more query types, Wong *et al.* [26] studied KNN computation of encrypted tuples stored at untrusted SP, and proposed a method supporting SP to calculate the relative distance between two encrypted data points. As virtual dimension is introduced, this method could not build up index of encrypted data points efficiently. Therefore, when dealing with query request, SP has to traverse all encrypted data points, which leads to relatively poor query efficiency. To achieve fine-grained access control, Yang *et al.* [27] proposed an efficient scheme with privacy-preserving policy in the big data outsourcing scenario. They hide the whole attribute names and values in the access policies. This scheme supports the data authorization on the basis of the consumers' attributes, but does not satisfy the requirements of the fine-grained spatial query authorization and integrity verification. Guo *et al.* [28] developed a KD-tree based indexing technique to support range searches on encrypted uncertain IoT data, which is outsourced to the cloud service provider. To reduce the computing and communication cost for mobile device users, Li *et al.* [29] proposed a scheme to offload the computation task from mobile devices to the cloud, and optimize the communications. All the above schemes are suitable for string or numeric values. However, the spatial data cannot be trivially transformed into string or numeric values because the relationship among spatial data is completely different from that between string or numeric values, thus these schemes cannot be directly applied to privacy protection of spatial data.

To enable secure query processing for outsourced encrypted spatial data, some researchers employed space filling curves to transform POIs, which could support range and KNN query in the transformed space. In practice, SHC is applied in most circumstances. By using SHC, Ni *et al.* [7] reduced extra work load due to users' parameters configuration. Meanwhile, the computation and communication cost of range query is reduced since the clustering and distance preserving properties of SHC was superior. To improve the accuracy and efficiency of spatial query, Khoshgozaran *et al.* [8] introduced dual Hilbert curve technique, and used SHC to map data points and query request to Hilbert transformed space and enabled query services on encrypted spatial data. To achieve better data confidentiality, Talha *et al.* [1] proposed a transformation based scheme, which performs the encrypted queries at the cloud service provider and returns the encrypted results to the user. This scheme employs the Hilbert curve to generate the index of the encrypted spatial data and then used the cryptographic primitives to enhance the security level. There are also some privacy-preserving spatial query applications in smart city, Ni *et al.* [30] proposed a Bloom filter-based data retrieval mechanism to support accessible parking spots query while preserving the drivers' privacy. Wang *et al.* [31] present a novel two level search scheme to protect the privacy of clients' spatial datasets stored and queried at a public server. To achieve secure and accurate geometric range query, Xu *et al.* [32] employ secure KNN computation, polynomial fitting technique, and order-preserving encryption over the cloud data. However, these schemes cannot support the fine-grained query capability authorizaiton, which is an urgent requirement of data owners. In addition, these schemes do not provide the integrity verification for the query result, so the data owner and consumers could not validate whether the cloud service provider is trustworthy.

2.2 Integrity Verification of Query Result

To support integrity verification for the outsourced data, several solutions [13], [14], [15], [16], [17], [18], [19], [20] have been proposed for rational database. Assuming all records are sorted, Pang *et al.* [13] employed aggregated signature to sign each record with the information of adjacent records. Sion *et al.* [14] introduced challenge token for a SP to provide a certification of user's query, then integrity verification of the query result is conducted at the client side. Devanbu *et al.* [15] utilized MHT to authenticate data records, which calculates a signature based on the MHT structure and distributes it to clients for correctness validation. Based on dynamic hash table, Tian *et al.* [33] present a public auditing scheme for secure cloud storage. The scheme achieves good updating efficiency and batch auditing. Ji *et al.* [34] presented a system to enable verifiable attribute-based search over shared cloud data. This system provides good search experience and supports authenticated query processing. Wang *et al.* [16] proposed an identity-based data outsourcing scheme, which equips with desirable features advantageous over existing proposals in securing outsourced data. The scheme supports the verification of both file origin and file integrity. Yu *et al.* [17] proposed a construction of identity-based protocol, which employed key-homomorphic cryptographic primitive to reduce the system complexity and the cost for establishing

TABLE 1
Notations

Symbol	Description
Φ	Original POI set
Φ'	Encrypted POI set
$In(\Phi')$	Indexed POI set with digest
p, p'	POIs belonging to Φ, Φ'
(x_c, y_c)	Sequence number pair in x-axis and y-axis for coordinate (x, y)
Ω	The spatial domain
R	A rectangular area in spatial domain
S_0	Space filling curve starting point
N	Space filling curve order
θ	Space filling curve orientation
Γ	Space filling curve scaling factor
Ψ	Transformation key tree for the spatial domain
G	Curve state transformation table
$H(m)$	Digest of message m, $H()$ is a collision resistant hash function
C	The maximum number of POIs that a partitioned region contains in DSC
d	Digest of a POI's information
$Sign(d)$	Data owner's signature on digest d
$Sign^{-1}(h_T)$	The decrypted digest signed by Data owner
V_D	POI's index value generated by DSC
r	User's authorized region
STK_i	Transformation key for the authorized region
S_k	The symmetric key for POI encryption
T	MQ-tree for query integrity verification

and managing the public key authentication framework. Yao *et al.* [18] investigated the verifiable social data outsourcing and proposed a scheme for verifiable queries over outsourced social data, which is purchased by the third-party social data provider, and consumers can verify the trustworthy of the data provider. Zhang *et al.* [35] designed a verification structure and proposed a deterrent-based scheme to achieve query results verification for secure ranked keyword query. This scheme keeps the cloud from knowing the relationship among data and verification information, thereby increasing the probability of discovering dishonest cloud providers. However, because the verification structure should contain the relationship among the spatial data, which is different from the relationship among string or numeric values, these schemes cannot directly support spatial query verification.

For spatial query verification, Yang *et al.* [36], [37] proposed MR-tree and MR*-tree, which are space-efficient authenticated data structures suitable for fast query processing and verification. A partially materialized digest scheme [38] verifies one-dimensional queries and applies to both static and dynamic databases. It employs separate indexes for the data and their associated verification information to avoid unnecessary query processing costs. However, these solutions do not consider privacy preservation and integrity verification of the outsourced spatial data simultaneously. Ku *et al.* [9] designed the SHC-based probabilistically replicating scheme, which employs SHC to transform the spatial data before outsourcing it to SP. Then, by probabilistically replicating a portion of the spatial data and transforming it with a different space encryption key, the authorized user can verify the integrity of the spatial query result. However, SPR is not a deterministic integrity verification method, thus the cloud may escape the auditing process, and SPR cannot achieve the fine-grained authorization for query verification. The scheme proposed in this paper supports the fine-grained authorization for query verification, and does not introduce false negative in the query result verification.

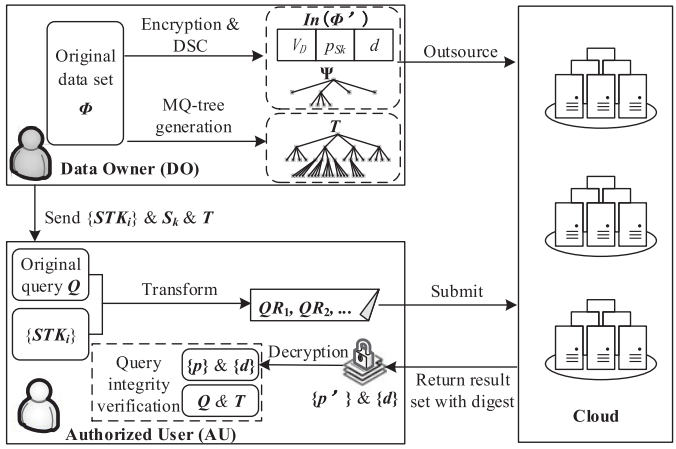


Fig. 1. System Model for secure spatial query with integrity verification.

3 Problem Statement

In this section, the system model and threat model are introduced and the design goal is identified for the fine-grained query authorization with integrity verification. Table 1 summarizes the notations used in this paper.

3.1 System Model

The system model is depicted in Fig. 1, including three entities, the data owner (DO), authorized user (AU), and cloud.

1) The data owner has tremendous volumes of location related data, namely, the original POI data. The data owner allows authorized users (i.e., third-party companies) to generate query tokens in their authorized region. The data owner outsources its data to the cloud for saving the costs of data storage and query services. For privacy concerns, the data owner will encrypt the data before outsourcing it to the cloud.
2) The authorized users are third-party companies that need the spatial data to do analysis and make business decisions. The authorized users should acquire the decryption key S_k, transformation key STK_i, and the verification structure T_i from the data owner in advance.
3) The cloud has sufficient computational and storage resources. To avoid the cloud from accumulating the intermediate information about the outsourced spatial data and users' queries, in the system model, the cloud stores the encrypted POI data for the data owner, and only provides the searching services for the authorized users.

Assuming the DO possess a collection of n POIs within a spatial domain Ω. Each POI p_i is represented as a tuple $< x, y, info >$, where x and y are its coordinates, while $info$ is the additional information about p_i, such as name, zipcode, and descriptive information, etc. Before outsourcing the dataset to the cloud, the DO chooses secret transformation parameters and builds query indexes of the POIs using DSC. For privacy preservation of the POI, the DO encrypts

$< x, y, info >$ with a symmetric key S_k, which is shared between the DO and AU. The structure of an encrypted POI is $< V_D, \{x, y, info\}_{S_k}, d >$, where V_D is the DSC-based query index of the POI, and digest d is computed as $d = H(x|y|info)$ (where | indicates concatenation).

The system model contains the data pre-processing and query processing phases. In data pre-processing phase, DO builds query indexes of POIs by DSC, and generates the indexed POI set $In(\Phi')$ for outsourcing, and then the DSC-based transformation key tree Ψ is generated. Note that the shared transformation key $STK_i \subseteq \Psi$, and the AU can only generate the query token in the authorized region r that STK_i corresponds to, which means the DO can distribute different transformation keys to different AUs to achieve fine-grained query capability authorization over the encrypted spatial data. Meanwhile, MQ-tree T is generated for verifying the integrity of the query results, and the nodes of T correspond to the same regions as the nodes of Ψ correspond to. Therefore, the proposed scheme also supports fine-grained verification authorization, which requires DO to share a verification structure T_i $(\subseteq T)$ that has the same authorized region r with STK_i.

In query processing phase, AU uses transformation key STK_i to transform the original query Q, and obtains a series of one-dimensional consecutive integer values $\{[l_j, h_j] | l_j \leq h_j\}$, which are denoted as query runs $\{QR_j\}$, namely, the query tokens. Then, AU submits $\{QR_j\}$ to cloud. Upon receiving the query result $\{p'\}$ and the corresponding digest $\{d\}$, AU decrypts $\{p'\}$ by S_k and gets the plaintext result set $\{p\}$. Finally, AU verifies the query result integrity based on the verification structure T_i, result set $\{p\}$, digest $\{d\}$, and the original query Q.

Compared with the system model in [1], the system model in this paper also includes three entities. However, to support fine-grained query and verification capabilities authorization, the model in this paper contains the transformation key tree generation and verification structure generation modules.

3.2 Threat Model

The cloud stores the outsourced spatial data and responds to the queries. However, the cloud would analyze the stored data and the AUs' query content to obtain valuable information about DOs and AUs. Moreover, the cloud may inject extra POIs or delete the competitors' top-ranking POIs for paid advertisers. Therefore, in this paper, it is assumed that the cloud is not fully trusted and would probably behave dishonestly. The threats models in this paradigm is summarized as follows.

1) Brute-force attack. The attacker tries every possible transformation key of the outsourced spatial data until the correct one is found. The difficulty of this attack depends on the key length. If the key has n bits, there are 2^n possible keys to try, so the complexity of a brute-force attack to find the transformation key is proportional to 2^n.
2) Location reconstruction attack. The attacker can access a limited number of the POIs' locations and their corresponding query indexes. With these known mapping information, the attacker may infer the locations of other unknown POIs.
3) Record deletion attack. The attacker deletes several records from the correct query result set and sends the incomplete result set to AU. This attack occurs when the cardinality of the result set is unknown, such as the spatial range query.
4) Record substitution attack. The attacker substitutes several correct records with other records, which are also selected from the outsourced spatial data, but not in the correct result set. This attack occurs when the cardinality of the result set is known, such as KNN query.

3.3 Design Goal

The design goal is to achieve fine-grained query authorization with integrity verification. Specifically, the following three objectives should be achieved.

1) Query privacy preservation. The locations of the DO's outsourced spatial data and the AU's spatial queries should be preserved to prevent privacy leakage. The proposed scheme should be resilient to the potential attacks.
2) Fine-grained authorization for query result verification. In AU's authorized region, it can deterministically verify the integrity of the spatial query result, which may be modified by the cloud for financial benefits. The proposed scheme should be capable of detecting the modification over the query result without false negative probability.
3) Computation and storage efficiency. The proposed scheme should consume low computational and storage resources in terms of query index and verification structure generation as well as the query result verification.

4 The Proposed Scheme

The proposed scheme includes two parts: data pre-processing and query integrity verification. First, the original locations of POIs are transformed from 2-dimensional coordinates to 1-dimensional query index values. Second, the transformation key tree Ψ and verification structure MQ-tree T are constructed, which enable DO to authorize the capabilities of query token generation and query result integrity verification to the AUs. Third, the AU can generate the query token and verify the query result integrity in his authorized region r, which is defined by the transformation key $STK_i \subseteq \Psi$ and the verification structure $T_i \subseteq T$. Note that STK_i and T_i correspond to the same region $r \subseteq \Omega$, and are transmitted by the DO to AU over a secure communication channel using Transport Layer Security (TLS) [39]. The TLS aims to provide secure communication using classical TCP sockets with very few changes in API usage of sockets to be able to leverage security on existing TCP socket code. The TLS is used in every browser worldwide to provide https (http secure) functionality, and can be used to exchange confidential information between the DO and AU.

4.1 Density-Based Space Filling Curve (DSC) for Data Pre-Processing

DSC is the building block of the secure and fine-grained spatial data pre-processing. It includes three phases: 1)

Quadtree-based partition [40] is employed over the spatial domain. The quadtree nodes are used to represent the atom regions, which cannot be further partitioned. 2) Spatial index vaule is generated based on fractal rules of Hilbert curve. Each atom region is traversed sequentially in accordance with the fractal rules of Hilbert curve, which is specified by the curve parameters given by the DO. Each atom region obtains a sequence number in the traversing process, and this number is denoted as DSC value, which is used as the query index of POI. 3) Transformation key tree is constructed for the fine-grained authorization of the query token generation. Based on DSC, the verification structure MQ-tree is generated for verifying the integrity of the spatial query result.

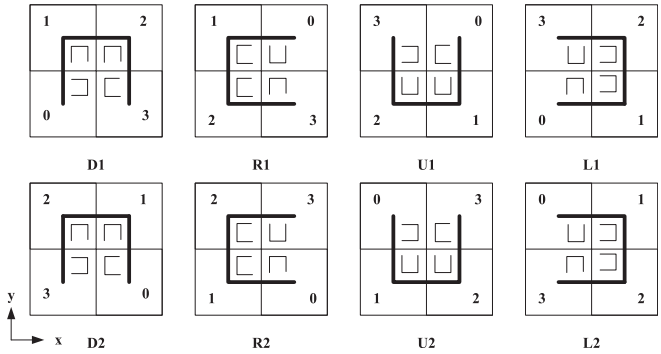


Fig. 2. Fractal rules of Hilbert curve.

4.1.1 Quadtree-Based Space Partition

In density-based partition, the partition granularity depends on the capacity, which is the maximum number of POIs an atom region contains, denoted as C. A small C value will lead to a fine-grained partition over the spatial domain, and vice versa.

The quadtree-based space partition results in a one-to-one correspondence between a quadtree node and a region, and the quadtree node is used to denote the corresponding region for simplicity. Algorithm 1 illustrates the generation process of the space quadtree. In this algorithm, Z_C denotes the child node set of a quadtree node Z, and Z_P denotes the POI set in the region of node Z. $Z(i)$ denotes the ith child node of Z, and E_L denotes the coordinate of POI E. The region corresponding to node Z is denoted by Z_R.

Algorithm 1. Space Quadtree Generation

Input: original POI set P, capacity C
Output: quadtree root node Q
1: create a root node Q
2: **for** each $p \in P$ **do**
3: QuadTreeInsert(p, Q, C)
4: **end for**
5: **return** Q
6: **function** QuadTreeInsert(p, Z, C)
7: **if** $Z_C = \emptyset$ **then**
8: **if** $|Z_P| < C$ **then**
9: $Z_P \leftarrow Z_P \cup p$
10: **else**
11: $Z_C \leftarrow \{Z(0), Z(1), Z(2), Z(3)\}$
12: **for** each $E \in Z_P$ **do**
13: **if** $E_L \in Z(i)_R$ **then**
14: $Z(i)_P \leftarrow Z(i)_P \cup E$
15: **end if**
16: **end for**
17: get $D \in Z_C$ which meets $p \in D_R$
18: QuadTreeInsert(p, D, C)
19: **end if**
20: **else**
21: get $D \in Z_C$ which meets $p \in D_R$
22: QuadTreeInsert(p, D, C)
23: **end if**
24: **end function**

This algorithm is performed on the DOs side, and composed of two procedures. Procedure QuadTreeInsert inserts a POI p to the quadtree node Z recursively. This procedure determines the inserting node of POI p according to p's coordinate, capacity C and the current quadtree node Z. If Z is a leaf node, and $|Z_C|$ is smaller than C, then p is directly inserted to node Z (lines 7-9). Otherwise four child nodes are created for node Z, and the POIs of node Z will be redistributed into its corresponding child nodes (lines 10-16). After that, this procedure affirms the child node that p belongs to, denoted as D (line 17). Then POI p is inserted to node D (line 18). If Z is an intermediate node, it first affirms the child node which p belongs to, denoted as D (lines 20-21). Then p is inserted to node D by calling procedure QuadTreeInsert recursively (line 22). The main procedure inserts each POI $p \in P$ to root node Q sequentially, and finally constructs the space quadtree structure (lines 1-4). Fig. 3a illustrates an example of the quadtree-based space partition. It can be seen that the partition continues until each atom region contains at most one POI. The algorithm inserts the POIs to the root node of the space quadtree, and splits the nodes that contain more than one POI. In this figure, each atom region corresponds to a leaf node of the space quadtree, and the whole region corresponds to the quadtree's root node.

If $|P| = n$, then the maximum number of generated leaf nodes is $4n/C$. Averagely, the depth of the quadtree is $log_4(4n/C) + 1$. On average, procedure QuadTreeInsert requires $log_4(4n/C) + 1$ comparisons, C POI redistributions, and 1 insertion when inserting a POI. Therefore, Algorithm 1 needs $\sum_{i=1}^{n}(log_4(4i/C) + 1 + C + 1) < n(log_4(4n/C) + 1 + C + 1)$ operations, meaning that the complexity of Algorithm 1 is $O(nlog_4 n)$.

4.1.2 Index Generation for DSC

According to the constructed quadtree structure, the DSC values of the leaf nodes and the sub-curve parameters (including sub-curve orientation and starting point) of the intermediate nodes are generated. Algorithm 2 shows the details of the DSC value generation, which executes on the DOs side. Because the POIs are distributed into the leaf nodes, the index value of each POI is set the same as the DSC value of the leaf node that the POI belongs to. Fig. 2 depicts the fractal rules of Hilbert curve that Algorithm 2 employs.

As Fig. 2 shows, a space filling curve with order N partitions the spatial region into $2^N \times 2^N$ cells, while the starting point and orientation denote where the curve begins and the open end direction of the curve, respectively. The fractal rules of Hilbert curve include eight types according to different curve orientations and starting points. The sequence number

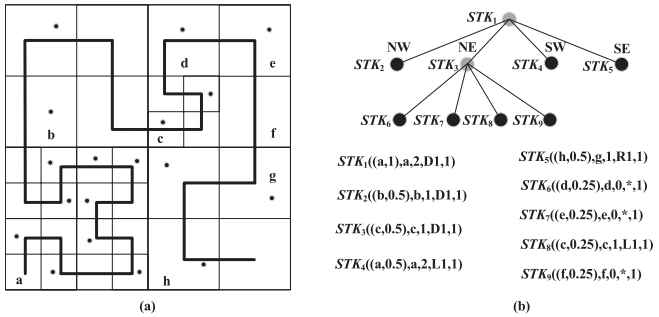


Fig. 3. (a) DSC, (b) Transformation key tree.

of sub-region is assigned by the number in the quad, and each sub-region can be further partitioned by the corresponding fractal rule. To facilitate processing, curve state transformation table (CSTT) is applied to represent the fractal rules. Table 2 shows CSTT for fractal rule D1, and other rules' CSTTs can be represented similarly. By looking up CSTT, the sequence number of sub-region (x_c, y_c) in current level, and CSTT in the next level are obtained.

In Algorithm 2, subscripts $_O$ and $_S$ denote the sub-curve's orientation and starting point for the corresponding tree node, respectively. Q_O is sub-curve orientation of node Q, Q_S is sub-curve starting point of node Q, S is a stack that stores quadtree nodes, G is CSTT, Z_D is DSC value of node Z, c is the counter of the DSC index value. According to the curve orientation θ and starting point S_0, Algorithm 2 traverses the space quadtree Q in depth-first order, and generates the DSC value of each leaf node according to the visiting sequence. The algorithm sets starting point and sub-curve orientation of each intermediate node in the top-down manner on the basis of the fractal rules (lines 7-11). In this algorithm, $G_O(Z_O, Z_S, i)$ and $G_S(Z_O, Z_S, i)$ denote the sub-curve orientation and starting point of node Z's ith child node, respectively. If D1 is set as the fractal rule of the DSC, then the DSC values of the atom regions b, c, d, e in Fig. 3a are 16, 20, 24, 25, respectively.

Algorithm 2. Index Generation for DSC (IGD)

Input: quadtree root node Q, starting point S_0, curve orientation θ
Output: updated quadtree root node Q

```
1: c ← 0
2: Q_O ← θ
3: Q_S ← S_0
4: S.PUSH(Q)
5: while S ≠ ∅ do
6:    Z ← S.POP()
7:    if Z_C ≠ ∅ then
8:       for i = 3; i ≥ 0; i − − then
9:          Z(i)_O ← G_O(Z_O, Z_S, i)
10:         Z(i)_S ← G_S(Z_O, Z_S, i)
11:         S.PUSH(Z(i))
12:      end for
13:   else
14:      Z_D ← c
15:      c ← c + 1
16:   end if
17: end while
```

TABLE 2
Curve State Transformation Table for D1

y_c \ x_c	0	1
0	0 / L1	3 / R1
1	1 / D1	2 / D1

If $|P| = n$, then Algorithm 2 visits $\sum_{i=0}^{log_4(4n/C)} 4^i = \frac{16n-C}{3C}$ nodes only once averagely. Therefore, its complexity is $O(n)$.

4.1.3 Transformation Key Tree Generation for DSC

The DSC values of all leaf nodes in the space quadtree are used to build query indexes of POIs. In order to improve the query efficiency and support the fine-grained authorization of spatial region, Algorithm 3 is designed to extract sub-curve parameters in sub-regions with different density, and then generate the transformation key tree Ψ for DSC. This algorithm is performed on the DOs side. In algorithm, Z_V is a flag to denote whether node Z is visited, Z_N and Z_F are sub-curve order and father node of node Z, respectively.

Algorithm 3. DSC Key Tree Generation

Input: quadtree root node Q
Output: updated quadtree root node Q

```
1: S.PUSH(Q)
2: while S ≠ ∅ then
3:    Z ← S.POP()
4:    if Z_C ≠ ∅ then
5:       if {Z_C}_V = TRUE then
6:          if {Z_C}_C = ∅ ∧ {Z_C}_N = constant then
7:             Z_N ← {Z_C}_N + 1
8:             Z_C ← ∅
9:             Z_V ← TRUE
10:         else
11:            Z_V ← TRUE
12:         end if
13:      else
14:         S.PUSH(Z)
15:         for i ← 3; i ≥ 0; i − − do
16:            S.PUSH(Z(i))
17:         end for
18:      end if
19:   else
20:      if Z_N > (Z_F)_N then
21:         (Z_F)_N ← Z_N
22:      end if
23:      Z_V ← TRUE
24:   end if
25: end while
```

Algorithm 3 prunes the quadtree nodes (can be rebuilt by the parameters of their father node) in bottom-up manner and updates the parameters of transformation keys of the remaining quadtree nodes. After that, the leaf node may correspond to a region that consists of several atom regions, and a new quadtree is generated. This quadtree is denoted as transformation key tree for DSC, as Fig. 3b depicts.

In this transformation key tree, the distribution of the sub-curve in the leaf node is uniform. The leaf node contains the transformation key $STK(R, S_0, N, \theta, \Gamma)$ of the

corresponding region, such as STK_4. The distribution of the sub-curve in the intermediate node is not uniform, which has four child nodes, denoted as NW, NE, SW, and SE. The intermediate node contains the transformation key $STK(R, S_0, N_M, \theta, \Gamma)$ of the corresponding region, where N_M is the highest curve order of all its child nodes, such as STK_3. The atom region does not contain any sub-curve, so the curve order and orientation of the corresponding node are 0 and null, respectively, such as STK_6. Note that Algorithms 1 and 2 have already generated the parameters R, S_0, and θ. Algorithm 3 prunes unnecessary bottom nodes and generates the parameter N. The DO provides the AU with the transformation key STK_i ($\subseteq \Psi$), which enables the AU to generate query tokens in the STK_i's corresponding region r ($\subseteq \Omega$).

Algorithm 3 visits each node twice. If $|P| = n$, then it requires $2\sum_{i=0}^{log_4(4n/C)} 4^i = \frac{32n-2C}{3C}$ visits. Thus, it has a complexity of $O(n)$.

Algorithm 4. MQ-Tree Generation (MQG)

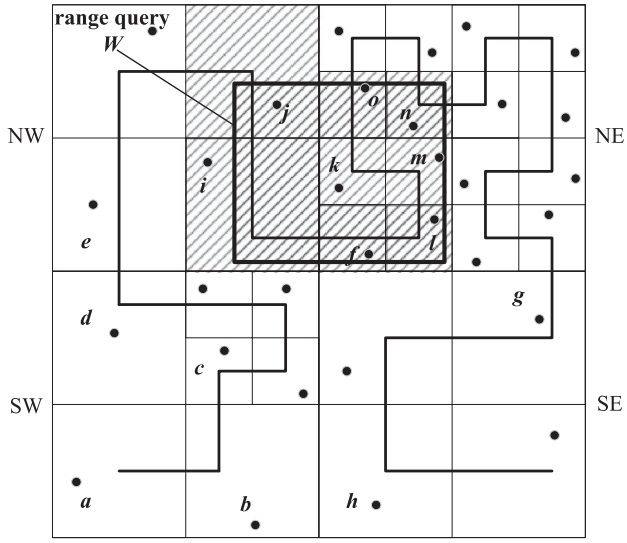
Input: space quadtree root node T
Output: MQ-tree root node T

1: S.PUSH(T)
2: **while** $S \neq \emptyset$ **do**
3: $Q \leftarrow$ S.POP()
4: **if** $Q_C \neq \emptyset$ **then**
5: **if** $\{Q_C\}_V =$ TRUE **then**
6: **if** $\{Q_C\}$ has digests **then**
7: $h_Q \leftarrow H(h_{Q(0)}|h_{Q(1)}|h_{Q(2)}|h_{Q(3)})$
8: $Q_V \leftarrow$TRUE
9: **end if**
10: **else**
11: S.PUSH(Q)
12: **for** $i \leftarrow 3; i \geq 0; i--$ **do**
13: S.PUSH($Q(i)$)
14: **end for**
15: **end if**
16: **else**
17: **if** Q does not contain POI **then**
18: $h_Q \leftarrow H(\$)$
19: **else**
20: $h_Q \leftarrow H(p_i.d|p_j.d|p_k.d)$
21: **end if**
22: $Q_V \leftarrow$ TRUE
23: **end if**
24: **end while**
25: $h_T \leftarrow Sign(h_T)$

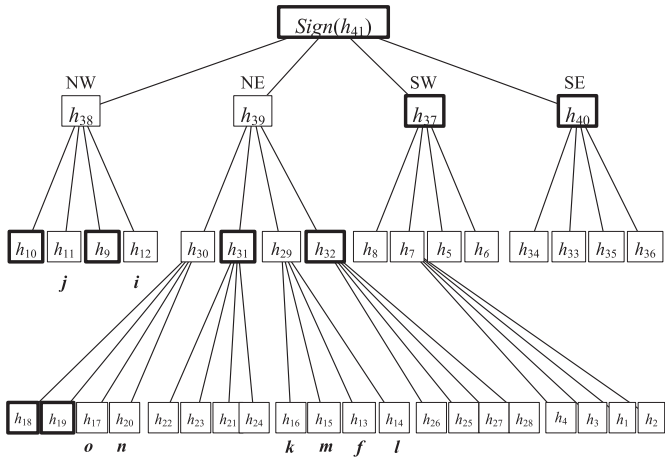
4.1.4 DSC-Based Verification Structure Generation

As an authenticated data structure, MHT authenticates a set of messages collectively, without signing each one individually [22]. Tuples are organized into a tree so that the signature to the root node can guarantee the data integrity of other nodes in the tree. The collision-resistance of the hash function guarantees that the cloud cannot modify any message in a way that leads to an identical root digest. It is possible to take the advantage of this feature and employ MQ-tree for integrity verification.

Based on the space quadtree generated by Algorithm 1, DO builds the MQ-tree upon the space quadtree's leaf nodes.



(a) Range query over DSC with $C = 1$



(b) Query result and MQ-tree

Fig. 4. Range query and verification example.

If the leaf node L does not contain any POI, then its digest is defined as $h_L = H(\$)$, where \$ denotes a constant string assigned by DO (line 18). If the leaf node L contains several POIs, then its digest is defined as $h_L = H(p_i.d|p_j.d|p_k.d)$, where $p_i.d \leq p_j.d \leq p_k.d$ (line 20). The MQ-tree is a quadtree, where each intermediate node Q is associated with a digest h_Q over the digests of its child nodes, and the concatenation order of these digests is according to the corresponding child nodes' index values (smaller index value first). The detailed process of generating MQ-tree is shown in Algorithm 4. In this algorithm, S is a stack for storing quadtree nodes, Q_C is the child node set of quadtree node Q, $Q(i)$ is the ith child node of Q, Q_V is a flag to denote whether node Q is visited.

Fig. 4 illustrates an example of DSC and its corresponding MQ-tree generation process. The atom region capacity C is 1, and the curve starting point is the lower left region. As Fig. 4b illustrates, the MQ-tree is generated in bottom-up manner, h_i is the digest of current node, and i denotes the concatenation order, for instance, $h_7 = H(h_1|h_2|h_3|h_4)$, and $h_{41} = H(h_{37}|h_{38}|h_{39}|h_{40})$, etc. The DO provides the AU with the verification structure T_i ($\subseteq T$), which enables the AU to verify the integrity of the query result in the T_i's corresponding region

r ($\subseteq \Omega$). Note that T_i is a sub tree of T, and the root node of T_i is labeled with $Sign(h_{T_i})$.

On average, Algorithm 4 visits $\sum_{i=0}^{log_4(4n/C)} 4^i = \frac{16n-C}{3C}$ tree nodes, and each node will be visited twice, meaning $2\sum_{i=0}^{log_4(4n/C)} 4^i = \frac{32n-2C}{3C}$ visits in total. Therefore, its complexity is $O(n)$.

Algorithm 5. Query Token Generation for Single POI

Input: POI p, transformation key STK_i
Output: p' query token Q_p

```
1: get Z ∈ STK_i which meets p ∈ Z_R
2: if Z = ∅ then
3:    return null
4: else
5:    compute (x_c, y_c) of p using Equation (2)
6:    G_0 ← G(Z_O, Z_S)
7:    offset ← 0
8:    for i ← Z_N − 1; i ≥ 0; i − − then
9:       offset ← offset << 2
10:      x_q ← x_c ∧ (1 << i)
11:      y_q ← y_c ∧ (1 << i)
12:      S_n/G_n ← G_0(x_q, y_q)
13:      G_0 ← G_n
14:      offset ← offset|S_n
15:   end for
16: end if
17: Q_p ← D_0 + offset
18: return Q_p
```

4.2 Query Token Generation and Result Verification

Based on the transformation key tree Ψ and verification structure MQ-tree T, the approaches are presented to generate the query token and verify the query result.

4.2.1 Query Token Generation for Single POI

The query token generation for a single POI is a basic function of spatial query processing. With the transformation key STK_i ($\subseteq \Psi$) provided by the DO, the AU can generate the query token of a POI p located in the authorized region r ($\subseteq \Omega$). Note that the authorized region r is defined by the transformation key STK_i. The details of the query token generation is shown in Algorithm 5. This algorithm is performed on AUs side.

With p's coordinate (x, y) and AU's transformation key STK_i, the algorithm first locates the region that p belongs to. If there is no such region, null will be returned, meaning that AU's authorized region does not contains p. Otherwise, a transformation key tree node Z ($\in STK_i$) will be obtained, and p belongs to region Z_R. (x_0, y_0) denotes the lower left endpoint of region Z_R, s is side length of region Z_R, Z_N is sub-curve order of node Z, and D_0 is the DSC value of node Z's sub-curve starting point. p's (x_c, y_c) in region Z_R should satisfy the following constraints.

$$\begin{cases} \frac{s}{2^N} \times x_c \le x - x_0 < \frac{s}{2^N} \times (x_c + 1) \\ \frac{s}{2^N} \times y_c \le y - y_0 < \frac{s}{2^N} \times (y_c + 1) \end{cases}. \quad (1)$$

With the above constraints, the values of x_c and y_c are obtained as follows.

$$\begin{cases} x_c = \lfloor \frac{x-x_0}{s} \times 2^N \rfloor \\ y_c = \lfloor \frac{y-y_0}{s} \times 2^N \rfloor \end{cases}. \quad (2)$$

With the current CSTT and p's (x_c, y_c), the sequence number S_n in the current level, and the next CSTT are obtained. Then the relative index value of p in region Z_R is calculated, which is denoted as $offset$ (lines 6-15). Finally, the query token of p in spatial domain is obtained by adding D_0 and $offset$.

If $|P| = n$ and AU has obtained the whole transformation key tree Ψ, then Algorithm 5 requires $log_4(4n/C) + 1$ comparisons for locating the POI and N loops for calculating the relative index value. Therefore, it needs $log_4(4n/C) + 1 + N$ operations in total, and the complexity of the algorithm is $O(log_4 n)$.

4.2.2 Query Token Generation for Range and KNN Queries

AU can generate the query tokens for a range query $W = [x_l, x_h] \times [y_l, y_h]$ with the transformation key STK_i. To reduce the communication overhead, the generated query tokens will be sorted and merged to 1-dimensional intervals, which are called query runs [41]. For simplicity, these query runs are also called as query tokens and denoted as $\{QR_i\}$. Note that there should be intersection between the AU's authorized region r and the range query W, or nothing will be generated. With the transformation key STK_i, the original range query W is decomposed into regions that the leaf nodes of STK_i correspond to. Then, the maximal block decomposition [41] and query runs calculation [42] are employed on each decomposed region to generate the query tokens of W. The detailed steps are shown as follows.

Step 1. From root node of STK_i, determine the region which the range query W belongs to. If there is no intersection between W and root node, quit; otherwise, continue.
Step 2. Determine intersection regions between W and each child node of current node, denoted as W_1, W_2, W_3 and W_4, note that these intersection regions may be empty.
Step 3. Repeat step 2 on each child node, until leaf node of STK_i is reached.
Step 4. Apply the maximal block decomposition and query runs calculation on all sub-regions corresponding to leaf nodes, and obtain the query runs.
Step 5. Sort and merge all query runs to generate the query token set $\{QR_i\}$.

The KNN query includes two types: approximate KNN query and accurate KNN query. For approximate KNN query, AU generates the query token Q_p of the given query POI p, then submits Q_p and K to the cloud. Since the query indexes of the outsourced spatial data are sorted in increasing order, the cloud can locate the query index qi that is closest to Q_p. Then, the cloud retrieves the encrypted data over the indexed POI set $In(\Phi')$ from the increasing and decreasing directions based on the index qi, until K results have been obtained, denoted as Re.

For accurate KNN query, AU first decrypts Re and gets POI q which is farthest from p, then generates a square region centered at p with side length $2 \parallel p - q \parallel$, where $\parallel$

$p - q \parallel$ denotes the Euclid distance between q and p. Using the above range query token generation approach, AU can obtain query result Re'. After decrypting Re', AU can get K nearest neighbor POIs to p as the accurate KNN query result Re''.

4.2.3 Integrity Verification of Range Query Result

Upon receiving the range query result set Re, the AU can verify the integrity of Re, as shown in Algorithm 6. This algorithm is performed on AUs side. $Q.V_D$ denotes the index value of the MQ-tree leaf node Q, and $W(i)$ is the ith sub-region of W and corresponds to $Q(i)$. $Re.keyset$ contains the index values of the result set Re, and Re.get$(Q.V_D)$ obtains the digest of the record from Re whose index value equals to $Q.V_D$. Note that if several POIs (e.g., p_i, p_j, p_k) have the same index value, then Re.get$(Q.V_D)$ will return $H(p_i.d|p_j.d|p_k.d)$ as the digest, where $p_i.d \leq p_j.d \leq p_k.d$.

Algorithm 6 traverses the MQ-tree in top-down manner and compares the tree nodes and sub-regions of the range query recursively. Finally, Algorithm 6 returns the verification digest d, and AU verifies whether d is signed by DO (using MQ-tree root node and DO's public key). Note that the AU's verification structure T_i and transformation key STK_i correspond to the same authorized region $r \subseteq \Omega$, and the integrity verification is valid only if there is an intersection between the authorized region r and range query W.

Algorithm 6. MQ-Tree Based Range Query Verification (MQRQV)

Input: Verification structure T_i, range query W, result set Re
Output: Verification digest d

$Q \leftarrow T_i$
if $Q_C = \emptyset$ **then**
  if $Q \cap W = \emptyset$ **then**
    $d \leftarrow h_Q$
  else
    if $Q.V_D \in Re.keyset$ **then**
      $d \leftarrow Re$.get$(Q.V_D)$
    else
      $d \leftarrow H(\$)$
    end if
  end if
else
  if $Q \cap W = \emptyset$ **then**
    $d \leftarrow h_Q$
  else
    for $i \leftarrow 0; i \leq 3; i++$ **then**
      $d \leftarrow d \mid \text{MQRQV}(Q(i), W(i), Re)$
    end for
    $d \leftarrow H(d)$
  end if
end if
return d

Fig. 4 illustrates an example for DSC-based range query and integrity verification process. Assuming that each atom region contains one POI ($C = 1$) and the AU has the transformation key tree Ψ and verification structure T, then the final space partition of DSC and the range query W are shown in Fig. 4a. For the range query W, the returned result set should be $\{j, i, f, l, m, k, o, n\}$, since their bounding regions (shown striped in Fig. 4a) overlap with W. The MQ-tree for integrity verification is shown in Fig. 4b, the bold box denotes the immediate obtained digest without accessing its child nodes. As the MQ-tree root node T has child nodes and partially overlap with W, the digest of the result set $d = H(h_{37}|H(h_9|h_{10}|H(j)|H(i))|H(H(H(f)|H(l)|H(m)|H(k))|H(H(o)|h_{18}|h_{19}|H(n))|h_{31}|h_{32})|h_{40})$. In this equation, h_i denotes the digest of current node and $H(x)$ denotes the digest of the node that contains POI x. The concatenation order is determined by the index values of the tree nodes. From a top view, the equation can be rewritten as $d = H(h_{37}|h_{38}|h_{39}|h_{40})$. Then, $h_{38} = H(h_9|h_{10}|H(j)|H(i))$ and $h_{39} = H(H(H(f)|H(l)|H(m)|H(k))|H(H(o)|h_{18}|h_{19}|H(n))|h_{31}|h_{32})$ are obtained. Then, AU can verify the integrity of the result by comparing d with $Sign^{-1}(h_T)$, if d equals with $Sign^{-1}(h_T)$, the result set is authentic and complete, otherwise, the cloud tampered the query result.

If $|P| = n$, in the worst case, Algorithm 6 requires $4(log_4(4n/C) + 1)$ comparisons. Thus, the complexity of Algorithm 6 is $O(log_4 n)$.

4.2.4 Integrity Verification of KNN Query Result

The integrity verification of the KNN query can be transformed into the integrity verification of the range query, which can be processed by Algorithm 6.

As Fig. 5a shows, given a query point q and $K = 3$, the accurate KNN query result set Re'' should be $\{j, i, k\}$. The corresponding range query W of the result set Re'' is shown as the bold box, and its result set Re' is $\{j, i, f, k, o\}$, it satisfies $Re'' \subseteq Re'$, and there is no POI in $\{f, o\}$ that is closer to p. So the verification process continues, and the digest of the result set Re' can be calculated by $d = \text{MQRQV}(T, W, Re')$. The MQ-tree for the KNN query integrity verification is shown in Fig. 5b, similar to Fig. 4, the bold box denotes the immediate obtained digest without accessing its child nodes. The digest of the result set $d = H(h_{37}|H(h_9|h_{10}|\ H(j)|H(i))|H(H(H(f)|h_{14}|h_{15}|H(k))|H(H(o)|h_{18}|h_{19}|h_{20})|\ h_{31}|h_{32})|h_{40})$. From a top view, this equation can be rewritten as $d = H(h_{37}|h_{38}|h_{39}|h_{40})$. Then, $h_{38} = H(h_9|h_{10}|H(j)|H(i))$ and $h_{39} = H(H(H(f)|h_{14}|h_{15}|H(k))|H(H(o)|h_{18}|h_{19}|h_{20})|h_{31}|\ h_{32})$ are obtained. Finally, the query result integrity can be verified by comparing d with $Sign^{-1}(h_T)$.

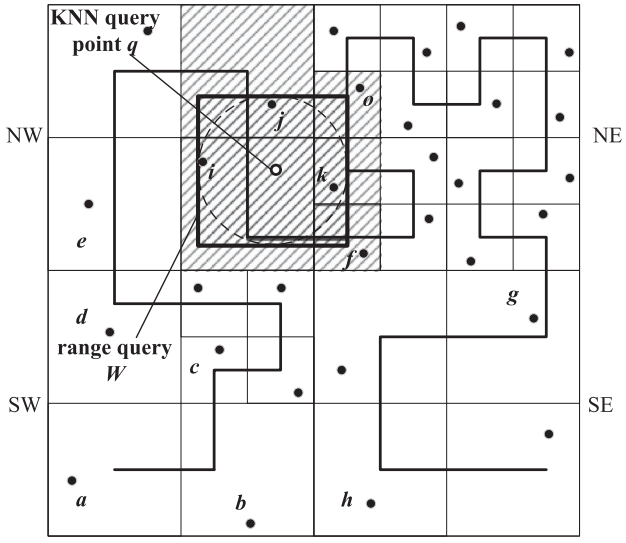
The complexity of the range query generation can be negligible, compared to the range query verification. Therefore, the complexity of the KNN query verification is $O(log_4 n)$.

5 Security Analysis

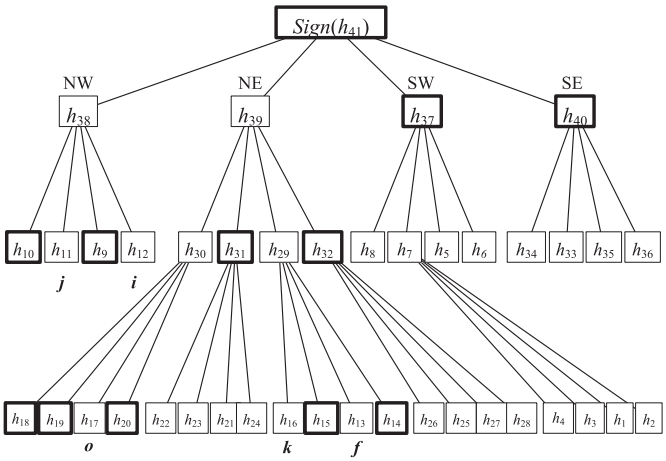
In this section, how the proposed scheme can resist all the attacks defined in Section 3.2 is discussed.

5.1 Brute-Force Attack

The transformation key tree Ψ consists of several transformation key $STK(R, S_0, N, \theta, \Gamma)$, and each STK includes the spatial region R, the curve starting point S_0, the curve order N, the curve orientation θ, and the curve scale factor Γ. Actually, the DSC can be treated as a combination of SHCs which vary in accordance with the density of POIs. Each SHC becomes a one-way function if the curve parameters are not known [11], so the attacker has to exhaustively



(a) KNN query over DSC with $C = 1$



(b) Query result and MQ-tree

Fig. 5. KNN query and verification example.

search all possible combinations of the curve parameters to find the right curve by comparing the query indexes of the POIs in the corresponding region, and the transformation key tree Ψ can only be cracked after finding the right curve parameters of all sub regions.

One sub region is taken to analyze the complexity of the brute-force attack. Because the curve starting point $S_0(x_0, y_0)$ should lie on the intersection of two lines that come from the X, Y axes separately. Although it is impossible to find the exact values of x_0 and y_0 in the continuous space, the attacker can approximately estimate x_0 and y_0 through generating finest partition on the region. In this way, the attacker can guarantee that his estimated starting point (x_0^*, y_0^*) is very close to (x_0, y_0), which satisfies $|x_0 - x_0^*| \leq \epsilon$ and $|y_0 - y_0^*| \leq \epsilon$. ϵ is a sufficiently small value that guarantees the generated query indexes of the POIs will not be affected if (x_0, y_0) is substituted by (x_0^*, y_0^*). To achieve this, the attacker has to search the region exhaustively. Let m denotes the number of bits for each coordinate. Hence, the attacker should enumerate 2^m values on each axis, and there are $2^m \times 2^m$ candidates in the searching space for the curve starting point. It is assumed s candidate values for the curve order N. The range of the curve orientation can be discretized to make sure the estimated curve orientation θ^* satisfies $|\theta - \theta^*| \leq \epsilon$. The attacker should search 2^n candidate orientation values assuming the number of bits is n. Similarly, the searching space for the curve scale factor is 2^k if the parameter is represented in k bits. The entire searching space in one region contains $2^m \times 2^m \times s \times 2^n \times 2^k$ candidate values. Assuming the attacker uses the similar number of bits (e.g., m bits) to represent the curve parameters, the complexity of the searching space in one region is $O(2^{4m})$ considering $s \ll 2^m$. Finally, the complexity of the brute-force attack is $O(v2^{4m})$, where v denotes the number of sub regions. Theoretically, the transformation key tree cannot be cracked in polynomial time. Therefore, DSC can resist brute-force attack.

5.2 Location Reconstruction Attack

In this attack, the attacker can access a limited number of the POIs' original coordinates L and their corresponding query indexes L'. With these known mapping information, the attacker may infer the coordinates of other unknown POIs. Because of the distance preserving properties of the space filling curves, the attacker can estimate the original coordinate of an unknown POI based on its query index and the known mapping information. By comparing the coherent pattern of feature vectors $V(p, L)$ and $V(k, L')$ with chosen candidate coordinate p and the corresponding query index k, the attacker is able to estimate the original coordinate of the given query index k with small error [43]. The $dissimilarity$ [43] is employed to measure the difference between $V(p, L)$ and $V(k, L')$. In this way, the attacker can reconstruct the original coordinates of the outsourced dataset. The $estimation\ distortion\ DT(D, D^*)$ [43] is employed to measure the average error between the reconstructed dataset D^* and the original dataset D. The SHC-based query index generation schemes partition the space with uniform guanularity, and the distribution of the generated query indexes shows the dense and sparse regions of the outsourced spatial data, which can be exploited by the attacker to reduce the estimation error. On the contrary, the distribution of the DSC-based query indexes are almost uniform, which leaks less distribution information compared with the SHC-based query indexes. The experimental analysis shows the DSC-based query indexes are more resilient against the location reconstruction attack. Note that the POIs belonging to the same atom region have the same query indexes, and the attacker can not distinguish the coordinates in the same atom region. Suppose δ is the minimal unit in the x-axis and l is the side length of the smallest atom region, then the probability of reconstructing the exact coordinate of an unknown index is $\frac{\delta^2}{\tau l^2}$, where τ is the number of candidate atom regions that the reconstructed coordinate belongs to. Because the τ value of SHC-based query index is less than that of DSC-based query index according to the above analysis, the probability of a successful reconstruction attack over the DSC-based query index is lower.

5.3 Record Deletion Attack

Suppose the cloud may modify or delete records from the correct query result, or insert other records that do not

belong to the outsourced spatial data. However, as each POI is encrypted and has a digest, modifying or inserting any record will be easily detected with out any false negative. It is because the collision-resistance of the hash function guarantees that forging or modifying a digest is impossible. Therefore, in this paper, two attacks against the integrity verification are considered [9]: 1) record deletion attack, the cloud delete several records from the correct query result set and send the incomplete result set to AU. This attack occurs when the cardinality of the result set is unknown, such as range query; 2) record substitution attack, cloud substitute several correct records with other records, which are also part of the outsourced spatial data, but not in the correct result set. This attack occurs when the cardinality of the result set is known, such as KNN query.

Assuming the correct range query result is Re, the cloud deletes one record from Re randomly and send the incomplete result set Re' to AU. As one record has been deleted, MQRQV cannot get its digest from Re' in the verification process, thus $H(\$)$ is taken as the verification digest, which is different from the correct digest. Based on the collision-resistance of the hash function, the finally generated digest d' of Re' cannot be equal with the correct digest d of Re. Thus, MQRQV can deterministically detect whether one record has been deleted from the correct result set. Similarly, if the cloud deletes two or more records from the correct result set, MQRQV can also detect it. Therefore, the integrity verification of the range query result is resilient to the record deletion attack.

5.4 Record Substitution Attack

With the accurate KNN query result set Re'' and the query point p, the corresponding range query W can be generated as a square region centered at p with side length $2 \parallel p - q \parallel$, where q denotes the POI farthest from p. Note that the generated range query W may introduce more POIs in the range query result set Re', which should satisfy $Re'' \subseteq Re'$, and there should be no POI in $Re' - Re''$ that its distance to p is more close than q. If $Re'' \nsubseteq Re'$, it confirms that the cloud has inserted POIs that do not belong to the correct result set. In addition, if there is POI in $Re' - Re''$ that its distance to p is closer than q, it means that the cloud has substituted the correct POI with other POI, either of the above two conditions will terminate the KNN query integrity verification process. Otherwise, the digest of Re' is derived by MQRQV(T_i, W, Re'), and the integrity of query result Re' is verified. Therefore, the integrity verification of the KNN query result can resist record substitution attack.

6 Experimental Evaluation

This section evaluates the computational, storage costs and the attack resistance property of the proposed query integrity verification scheme.

6.1 Datasets and Parameters Setting

The efficiency of the proposed query integrity verification scheme is evaluated through extensive experiments on real and synthetic datasets. The real datasets include 175, 813 POIs in North America (NA), 123, 593 POIs in North East USA (NE), 6, 105 POIs in Oldenburg (OL), 174, 956 POIs in San Francisco (SF), and 18, 263 POIs in San Joaquin County (TG) [44]. The synthetic datasets include: 1) a dataset under uniform distribution (UN, 100, 000 POIs), which is generated by random generator and 2) a skewed dataset (SK, 100, 000 POIs), which contains four Gaussian clusters (with $\sigma = 0.05$ and randomly chosen centers) and 99 percent of the POIs are included in these clusters, and the rest 1 percent of the POIs are uniformly distributed. These original datasets are pre-processed and normalized to the unit square $[0, 1]^2$ before applied to the experiments. The starting point, curve orientation and scale factor of the SHC are set as (0, 0), D1 and 1, respectively. AES-128 [45] is used to encrypt the POIs, SHA-256 [45] is employed to compute the digests of POIs and quadtree nodes, and RSA-1024 [45] is used to generate the signature of the root node's digest. The experiments are conducted on a workstation with Intel Xeon E3-1505M 2.8 Ghz and 16 GB RAM.

TABLE 3
Parameters Setting

Dataset	$\lambda = 1$		$\lambda = 1.4$	
	N	C	N	C
NA	13	1	11	2
NE	12	1	10	2
OL	11	1	8	2
SF	12	1	11	2
TG	12	1	9	2
SK	12	1	11	2
UN	11	1	9	2

In the experiments, the partition granularity λ, the average number of POIs contained in an atom region, are employed to measure the quality of the query index. Since SHC and DSC employ curve order N and capacity C in the space partition, respectively, the relationship between λ and the curve parameters of SHC and DSC are first studied. Table 3 shows the values of curve order N and capacity C on each dataset when applying λ equals to 1 and 1.4. The following experiments use this parameters setting, and 40 percent duplication rate (same with [9]) is taken for SPR.

6.2 Query Authorization Performance

6.2.1 Computational Cost of Query Index Generation

To support the secure spatial query, the spatial transformation methods generate the query indexes of the outsourced spatial data. The index generation performance of SHC and DSC is evaluated with parameters $\lambda = 1$ and 1.4, respectively. EDHO [46] and BIA [47] are employed to generate the SHC-based indexes of the outsourced datasets, and the proposed IGD is used to generate the DSC-based indexes. The index generation time is shown in Table 4.

The index generation time drops as λ increases, and more time is required for processing datasets NA, NE, SF, and SK than dataset UN. It is because the distributions of these real datasets are more dense, which require fine-grained space partition and thus increase the computational cost of index generation. With parameters $\lambda = 1$ and 1.4, on average, the index generation time of IGD is only 61, 43 percent of that by BIA, respectively. In contrast with the uniform space

TABLE 4
Index Generation Time (ms)

Dataset	$\lambda = 1$			$\lambda = 1.4$		
	EDHO	BIA	IGD	EDHO	BIA	IGD
NA	1,585	1,192	998	1,414	1,033	528
NE	1,065	766	484	891	648	284
OL	45	30	11	30	20	7
SF	1,475	1,092	681	1,361	986	415
TG	139	103	59	111	77	30
SK	849	612	389	783	560	249
UN	829	599	379	681	509	235

TABLE 6
Verification Structure Generation Time (ms)

Dataset	$\lambda = 1$		$\lambda = 1.4$	
	SPR	MQG	SPR	MQG
NA	18,920	4,491	18,832	2,924
NE	12,214	2,358	12,124	1,723
OL	693	114	678	79
SF	18,300	3,521	17,725	2,502
TG	1,928	375	1,911	265
SK	10,417	1,775	10,247	1,229
UN	10,280	2,013	10,187	1,432

partition of the SHC, the DSC employs different partition granularity for sub-regions with different densities, which means the sub-regions with less POIs requires coarse-grained space partition, and the computational cost of the index generation is thus reduced.

6.2.2 Overheads of Transformation Key Tree Generation

Since SHC-based scheme employs the uniform space partition, its transformation key can be used to generate the query tokens in the whole spatial region. To enable fine-grained query capability authorization, the proposed scheme constructs the transformation key tree, whose node contains the transformation key corresponding to the partitioned spatial region. The transformation key tree generation performance of the proposed scheme is evaluated with parameters $\lambda = 1$ and 1.4, respectively.

Table 5 shows the key tree generation time of the proposed scheme over different datasets. The generation time is proportional with the size of datasets, and more time is required when setting $\lambda = 1$. Dataset with more POIs will generate a space quadtree with more tree nodes, and thus increase the generation time of the transformation key tree. Moreover, the smaller value of λ means the partition granularity of the spatial region is fine-grained, and Algorithm 3 will take more time to prunes the quadtree nodes and thus generate the transformation key tree.

6.3 Integrity Verification Performance

6.3.1 Computational Cost of Verification Structure Generation

The computational cost of verification structure generation is an important metric for evaluating the spatial query integrity verification. It is set $\lambda = 1$ and 1.4 for SPR and the proposed scheme, and compare their time costs of generating verification structure in different datasets. MQG is the verification structure generation algorithm in the proposed scheme. The experiment is performed for 100 times and the verification structure generation time (ms) is averaged.

TABLE 5
Transformation Key Tree Generation Time (ms)

Dataset	$\lambda = 1$	$\lambda = 1.4$
NA	103	35
NE	35	22
OL	4	2
SF	49	32
TG	7	3
SK	37	17
UN	31	15

Table 6 shows the verification structure generation time of SPR and the proposed scheme over different datasets. The generation time of SPR and MQG drops as λ increases, and SPR consumes more time to generate the verification information of the replicated spatial data. Specifically, when setting $\lambda = 1$ and 1.4, on average, the computational cost of SPR is 4.26 and 6.4 times higher than that of MQG, respectively. SPR generates $r * n$ replicated records (r denotes the duplication rate), which introduce extra computational costs for indexes, digests, and signatures generation. On the contrary, MQG builds the MQ-tree on the basis of the space quadtree generated by DSC, and only tree nodes' digests generation and root node signature calculation are needed. Thus, the computational cost of MQG is significantly lower. Higher λ means lower curve order for SHC and fewer tree nodes for DSC, and thus reduces the computational costs of SPR and MQG.

6.3.2 Computational Cost of Integrity Verification

The computational costs of range/KNN query integrity verification of SPR and the proposed scheme are evaluated over different datasets. The verification time for 100 range/KNN queries is averaged to compare the computational costs of SPR and the proposed scheme. The results for range query verification and KNN query verification using dataset NA are reported in Figs. 6 and 7, respectively. The results for other datasets are similar and omitted.

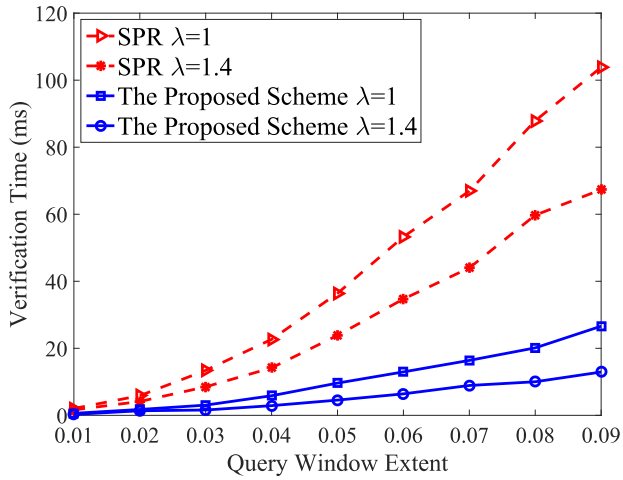


Fig. 6. Verification time of range query over NA.

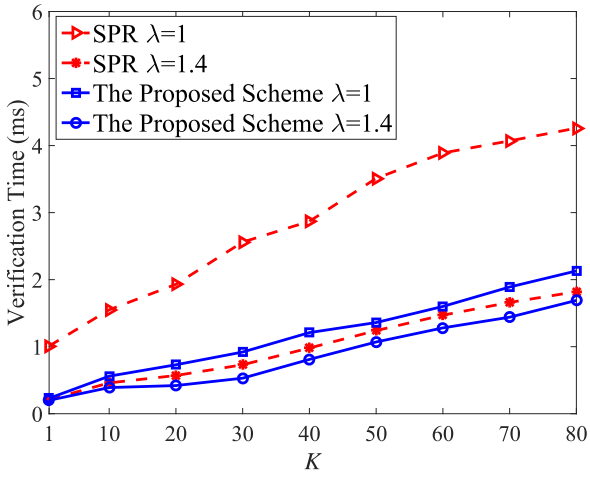


Fig. 7. Verification time of KNN query over NA.

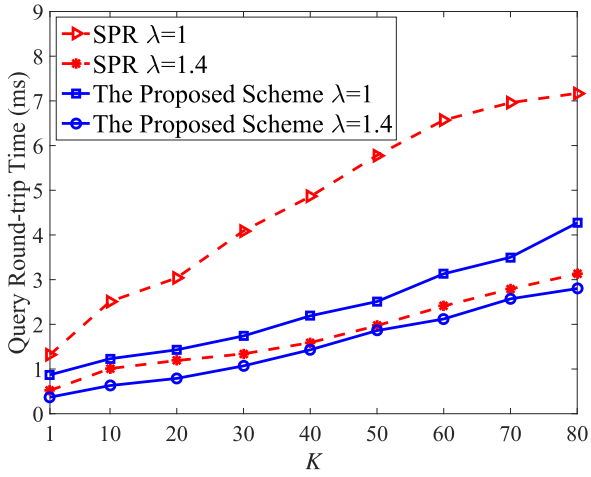


Fig. 9. Round-trip time of KNN query over NA.

As Fig. 6 shows, the proposed scheme requires less time on range query verification than SPR when choosing the same value of λ and query window extent. It is because SPR needs to decrypt the original query result and auditing query result, and then compare the two results to identify the integrity. The proposed scheme only needs to generate the digest with the original query result and MQ-tree, and verify whether the digest is signed by the DO. As the query window extent increases, the verification time of SPR grows quickly, while the time cost of the proposed scheme grows slowly. Even when the query window extent is 0.09 and $\lambda = 1$, the proposed scheme can also complete one range query verification in 26 ms.

Fig. 7 shows the verification time of SPR and the proposed scheme for a KNN query over dataset NA. Similar with the range query verification, the time cost of the proposed scheme is less than that of SPR. The verification time of SPR rises rapidly when $\lambda = 1$, while the time cost of the proposed scheme increases slightly. The proposed scheme and SPR transform KNN query verification to range query verification, and the transformed query window extent expands with the increasing of K. However, the verification time of KNN query is very little, because the transformed query window extent is very short for small value of K.

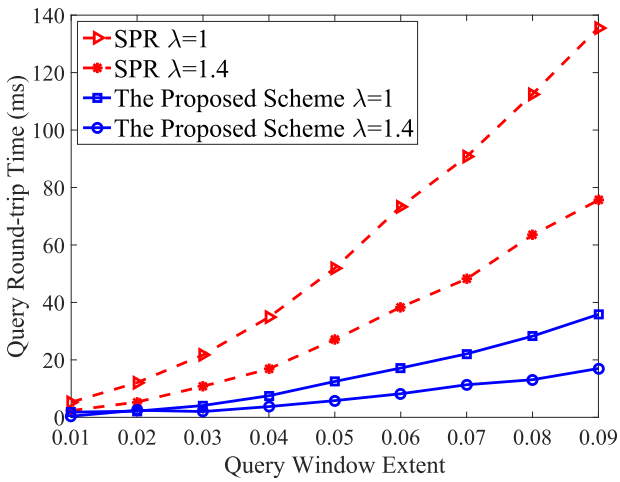


Fig. 8. Round-trip time of range query over NA.

6.3.3 Round-Trip Time of Spatial Query

The round-trip time of range/KNN query of SPR and the proposed scheme is compared over dataset NA. The round-trip time of range and KNN queries is shown in Figs. 8 and 9.

Compared with SPR, the proposed scheme generates less query runs and smaller size of result set, thus consumes less round-trip time on range query. Similar with the verification time evaluation, in Fig. 8, the round-trip time of SPR grows faster than that of the proposed scheme. The similar result can be found in Fig. 9 since the KNN query will be transformed to range query. The round-trip network delay time is not shown in the figure as it is similar for different query sizes.

6.3.4 Communication Cost of Range Query

The communication cost of range query of SPR and the proposed scheme is evaluated over dataset NA. Fig. 10 shows the amount of data that are exchanged between the cloud and AU. As bigger query window extent introduces more query runs in the range query transformation, the communication cost of SPR and the proposed scheme rises as the query windows extent increases. Moreover, with the partition granularity λ decreasing, the communication cost rises, and the cost of SPR increases faster than that of the proposed scheme. It is because when choosing small λ value, the generated atom regions of SPR will become much more than that of the proposed scheme, and thus the query runs generated by SPR will increase dramatically.

6.3.5 Storage Cost of Integrity Verification

In this section, the storage costs of the proposed scheme and SPR are compared over different datasets. For the proposed scheme, a MQ-tree is generated for query integrity verification and occupies the overhead storage, for SPR, duplicate POIs also consume extra storage space. It is assumed that the outsourced dataset contains a set of POIs, each with index value (4 bytes), spatial coordinates (8 bytes), and attached information (100 bytes).

Fig. 11 depicts the storage costs over different datasets. The storage costs of SPR are proportional to the size of the dataset, bigger λ cannot reduce the storage costs. However, the storage costs of the proposed scheme drop as λ

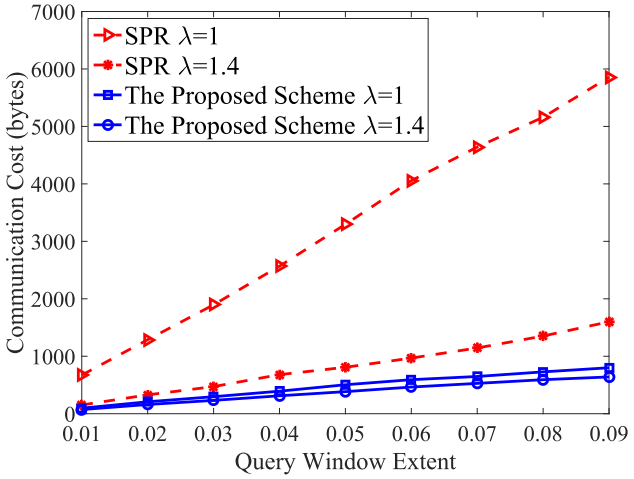
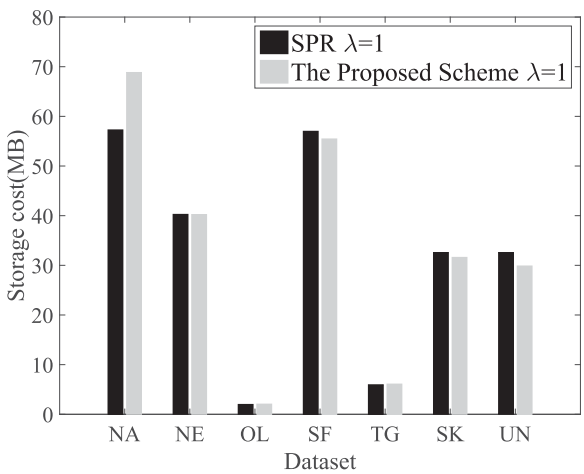
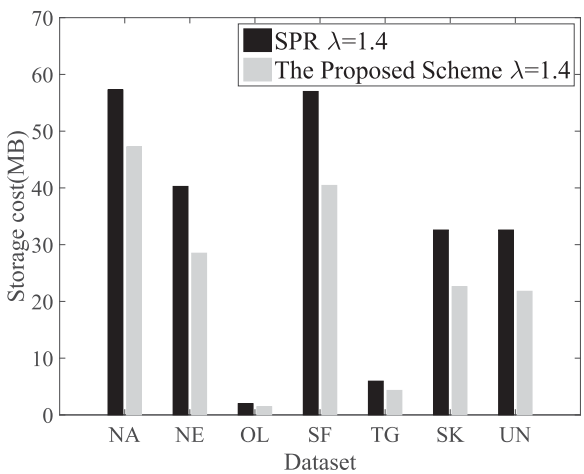


Fig. 10. Communication cost of range query over NA.

increases. On average, the storage costs of the proposed scheme are 2.9 percent higher than that of SPR when $\lambda = 1$, but when $\lambda = 1.4$, the costs of the proposed scheme are 26.9 percent lower than that of SPR. It is because the storage costs of SPR are determined by the duplication rate, while the storage costs of the proposed scheme are decided by the number of MQ-tree nodes, which decreases if bigger λ is selected. For SPR, the DO needs to choose a big enough duplication rate for reducing the false negative in the results verification, which reduces the efficiency of the query processing and integrity verification. Note that there may exist false negative in the results verification by SPR, it occurs if the POIs in the query region do not have the duplicate copies. Nevertheless, the proposed scheme introduces no false negative, because any tamper of the query result will lead to a wrong verification digest, which is different from the signed root node of MQ-tree. The results in Section 6.3.2 confirm that the proposed scheme consumes less computational cost of range/KNN query integrity verification than that of SPR. In consideration of the verification correctness and computational cost, the storage cost of the proposed scheme is acceptable.



(a) Storage Cost with $\lambda = 1$



(b) Storage Cost with $\lambda = 1.4$

Fig. 11. Storage cost over different datasets.

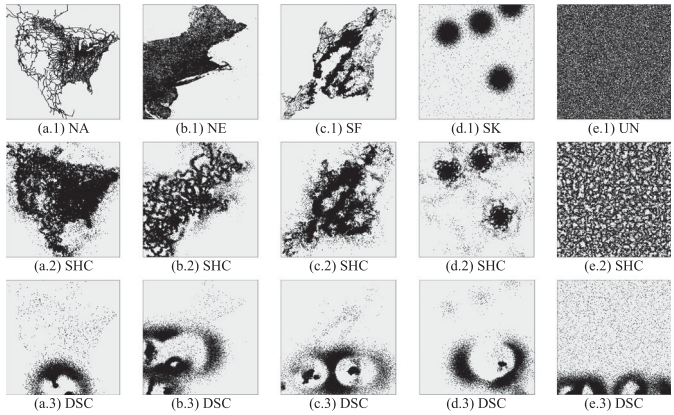


Fig. 12. Reconstructed coordinates datasets over the SHC-based indexes and DSC-based indexes.

6.4 Reconstruction Attack Evaluation

The location reconstruction attack model is employed to estimate the original coordinates of the outsourced datasets and visualize the attack effects against SHC-based and DSC-based schemes. The parameter λ is set to be 1. As the attacker is assumed to gain limited mappings between the original coordinates and the corresponding query indexes, the ratio of the known coordinates subset L is set to 1 percent of the outsourced spatial dataset. The reconstructed coordinates are illustrated in Fig. 12. The datasets NA, NE, SF, SK and UN are represented by a, b, c, d, and e, respectively. The original coordinates datasets, the reconstructed coordinates datasets over SHC-based indexes and DSC-based indexes are denoted by 1, 2, and 3, respectively. It is clear that the reconstructed coordinates datasets over the DSC-based indexes are less similar with the original coordinates datasets, which indicates the DSC-based indexes are more resilient to this attack.

Moreover, the numerical analysis is conducted over the original coordinates datasets and the reconstructed coordinates datasets by comparing the *estimation distortions* [43]. The value in Table 7 denotes the average error between the reconstructed dataset and the original dataset, and bigger value means better attack resistance property. Table 7 shows that the DSC-based indexes are more resilient to the reconstruction attack, comparing with the SHC-based

TABLE 7
Estimation Distortion

Dataset	SHC	DSC
NA	0.2473	0.3706
NE	0.2547	0.3236
SF	0.2103	0.3418
SK	0.2236	0.3941
UN	0.2861	0.4015

indexes. As the SHC-based index generation schemes employ high curve order in the sub regions with a small number of POIs, the distribution of the SHC-based indexes is not uniform, which provides more information about the distribution of the original data, thus the attacker can choose more accurate neighboring coordinates as references and generate the reconstructed coordinates dataset with smaller *estimation distortion* value.

7 Conclusion

In this paper, a fine-grained query authorization scheme with integrity verification is proposed over the encryped spatial data for location-based services. Considering the distribution of the spatial data, a density-based space filling curve is designed to generate the query indexes of the encrypted spatial data, and query token generation and result verification approaches are introduced to guarantee fine-grained and verifiable spatial query. The proposed scheme enables the data owner to achieve fine-grained spatial region authorization in both the query token generation and query result verification. Experimental results demonstrate that the computational cost of the index and verification structure generation approaches is less than that of SHC-based approaches, and the computational and storage costs of the integrity verification approach are less than that of SPR. In addition, the integrity verification scheme does not introduce false negative in the results verification. In the future work, the time factor will be considered in the fine-grained verifiable query authorization, which enables user to generate query tokens and verify the query results only in his authorized region and time range.

Acknowledgments

This work was supported in part by NSFC under Grant No. 61602290, No. 61672334, No. 61802242, and No. 61802241, in part by Natural Science Basic Research Program of Shaanxi Province under Grant No. 2017JQ6038 and No. 2020JM-288, in part by Foundation of Guizhou Provincial Key Laboratory of Public Big Data under Grant No. 2018BDKFJJ004, in part by Major Scientific and Technological Special Project of Guizhou Province under Grant No. 20183001, and in part by Xi'an Key Laboratory of mobile edge computing and security under Grant 201805052-ZD3CG36, and NSERC, Canada.

References

[1] A. M. Talha, I. Kamel, and Z. A. Aghbari, "Facilitating secure and efficient spatial query processing on the cloud," *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 988–1001, Oct.–Dec. 2019.

[2] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Providing task allocation and secure deduplication for mobile crowdsensing via fog computing," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 3, pp. 581–594, May/Jun. 2020.

[3] API by foursquare. 2018. [Online]. Available: https://developer.foursquare.com/places-api

[4] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1187–1198, Apr. 2016.

[5] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 772–785, Sep./Oct. 2019.

[6] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, 2000, pp. 44–55.

[7] W. Ni, J. Zheng, and Z. Chong, "HilAnchor: Location privacy protection in the presence of users' preferences," *J. Comput. Sci. Technol.*, vol. 27, no. 2, pp. 413–427, 2012.

[8] A. Khoshgozaran, H. Shirani-Mehr, and C. Shahabi, "Blind evaluation of location based queries using space transformation to preserve location privacy," *Geoinformatica*, vol. 17, no. 4, pp. 599–634, 2013.

[9] W. S. Ku, L. Hu, C. Shahabi, and H. Wang, "A query integrity assurance scheme for accessing outsourced spatial databases," *Geoinformatica*, vol. 17, no. 1, pp. 97–124, 2013.

[10] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz, "Analysis of the clustering properties of the hilbert space-filling curve," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 1, pp. 124–141, Jan./Feb. 2001.

[11] A. Khoshgozaran and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," in *Proc. 10th Int. Symp. Spatial Temporal Databases*, 2007, pp. 239–257.

[12] M. U. Arshad, A. Kundu, E. Bertino, A. Ghafoor, and C. Kundu, "Efficient and scalable integrity verification of data and query results for graph databases," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 866–879, May 2018.

[13] H. Pang, A. Jain, K. Ramamritham, and K. Tan, "Verifying completeness of relational query results in data publishing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 407–418.

[14] R. Sion, "Query execution assurance for outsourced databases," in *Proc. 31st Int. Conf. Very Large Data Bases*, 2005, pp. 601–612.

[15] P. T. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine, "Authentic third-party data publication," in *Proc. 14th Annu. Work. Conf. Database Secur.*, 2001, pp. 101–112.

[16] Y. Wang, Q. Wu, B. Qin, W. Shi, R. Deng, and J. Hu, "Identity-based data outsourcing with comprehensive auditing in clouds," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 940–952, Apr. 2017.

[17] Y. Yu *et al.*, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[18] X. Yao, R. Zhang, Y. Zhang, and Y. Lin, "Verifiable social data outsourcing," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[19] A. Yang, J. Xu, J. Weng, J. Zhou, and D. S. Wong, "Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2018.2851256.

[20] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, "Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 3, pp. 676–688, Mar. 2017.

[21] F. Tian, X. Gui, J. An, P. Yang, and X. Zhang, "A density-based space filling curve for location privacy-preserving," in *Proc. 11th IEEE Int. Conf. Services Comput.*, 2014, pp. 131–138.

[22] R. C. Merkle, "A certified digital signature," in *Proc. 9th Annu. Int. Cryptology Conf.*, 1989, pp. 218–238.

[23] H. Hacigümüs, B. Iyer, and S. Mehrotra, "Providing database as a service," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 29–38.

[24] H. Hacigümüs, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2002, pp. 216–227.

[25] R. Agrawal, J. Kiernan, R. Srikant, and Y. R. Xu, "Order-preserving encryption for numeric data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2004, pp. 563–574.

[26] W. Wong, D. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. Int. Conf. Manage. Data and 28th Symp. Princ. Database Syst.*, 2009, pp. 139–152.

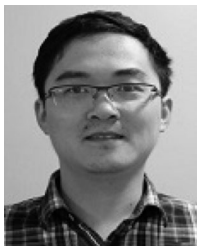
[27] K. Yang, Q. Han, H. Li, K. Zheng, Z. Su, and X. Shen, "An efficient and fine-grained big data access control scheme with privacy-preserving policy," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 563–571, Apr. 2017.

[28] C. Guo, R. Zhuang, Y. Jie, K. Choo, and X. Tang, "Secure range search over encrypted uncertain IoT outsourced data," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1520–1529, Apr. 2019.
[29] J. Li, R. Ma, and H. Guan, "TEES: An efficient search scheme over encrypted data on mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 126–139, Jan.–Mar. 2017.
[30] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Privacy-preserving smart parking navigation supporting efficient driving guidance retrieval," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6504–6517, Jul. 2018.
[31] B. Wang, M. Li, and L. Xiong, "FastGeo: Efficient geometric range queries on encrypted spatial data," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 2, pp. 245–258, Mar./Apr. 2019.
[32] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 870–885, Apr. 2019.
[33] H. Tian *et al.*, "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 701–714, Sep./Oct. 2017.
[34] Y. Ji, C. Xu, J. Xu, and H. Hu, "vABS: Towards verifiable attribute-based search over shared cloud data," in *Proc. 35th Int. Conf. Data Eng.*, 2019, pp. 2028–2031.
[35] W. Zhang, Y. Lin, and Q. Gu, "Catch you if you misbehave: Ranked keyword search results verification in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 74–86, Firstquarter 2018.
[36] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Spatial outsourcing for location-based services," in *Proc. 24th Int. Conf. Data Eng.*, 2008, pp. 1082–1091.
[37] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Authenticated indexing for outsourced spatial databases," *VLDB J.*, vol. 18, no. 3, pp. 631–648, 2009.
[38] K. Mouratidis, D. Sacharidis, and H. Pang, "Partially materialized digest scheme: An efficient verification method for outsourced databases," *VLDB J.*, vol. 18, no. 1, pp. 363–381, 2009.
[39] The transport layer security (TLS) protocol version 1.2. 2008. [Online]. Available: https://tools.ietf.org/html/rfc5246
[40] H. Samet, "The quadtree and related hierarchical data structures," *ACM Comput. Surv.*, vol. 16, no. 2, pp. 187–260, 1984.
[41] Y. Tsai, K. Chung, and W. Chen, "A strip-splitting-based optimal algorithm for decomposing a query window into maximal quadtree blocks," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 4, pp. 519–523, Apr. 2004.
[42] K. Chung, Y. Tsai, and F. Hu, "Space-filling approach for fast window query on compressed images," *IEEE Trans. Image Process.*, vol. 9, no. 12, pp. 2109–2116, Dec. 2000.
[43] M. Yiu, G. Ghinita, C. S. Jensen, and P. Kalnis, "Enabling search services on outsourced private spatial data," *The VLDB J.*, vol. 19, no. 3, pp. 363–384, 2010.
[44] Real datasets for spatial databases: Road networks and points of interest. 2009. [Online]. Available: https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm
[45] Java SE security. 2008. [Online]. Available: https://www.oracle.com/java/technologies/javase/javase-tech-security.html
[46] X. Liu and G. Schrack, "Encoding and decoding the hilbert order," *Softw. - Pract. Experience*, vol. 26, no. 12, pp. 1335–1346, 1996.
[47] C. Faloutsos and S. Roseman, "Fractals for secondary key retrieval," in *Proc. 8th ACM SIGACT-SIGMOD-SIGART Symp. Princ. Database Syst.*, 1989, pp. 247–252.

Feng Tian received the BS, and PhD degrees from Xi'an Jiaotong University, Xi'an, China, in 2009 and 2015, respectively. From 2018 to 2019, he was a visiting scholar with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is an assistant professor with the School of Computer Science, Shaanxi Normal University, China. His research interests include data privacy in cloud computing and network security.

Zhenqiang Wu received the BS degree from Shaanxi Normal University, Xi'an, China, in 1991, and the MS and PhD degrees all from Xidian University, China, in 2002, and 2007, respectively. He is currently a full professor of Shaanxi Normal University, China. His research interests include computer communications networks, mainly wireless networks, network security, anonymous communication, and privacy protection etc. He is a member of ACM and senior of CCF.

Xiaolin Gui received the BE, ME, and PhD degrees from Xi'an Jiaotong University, Xi'an, China, in 1988, 1993, and 2001, respectively, where he is currently a professor and the deputy dean of the School of Electronic and Information Engineering. His research interests include secure computation, data privacy, and the Internet of Things.



Jianbing Ni (Member, IEEE) received the PhD degree in electrical and computer engineering from the University of Waterloo, Waterloo, Canada, in 2018. He is currently an assistant professor with the Department of Electrical and Computer Engineering, Queen's University, Kingston, Canada. His research interests are applied cryptography and network security, with current focus on edge computing, mobile crowdsensing, Internet of Things, and blockchain technology.

Xuemin (Sherman) Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers university, New Brunswick, New Jersey, in 1990. He is currently a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular ad hoc and sensor networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, a Chinese Academy of Engineering Foreign fellow, and a distinguished lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the R.A. Fessenden Award, in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) presents, in 2019, James Evans Avant Garde Award, in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award, in 2015 and Education Award, in 2017 from the IEEE Communications Society, and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award, in 2006 from the University of Waterloo, Canada and the Premier's Research Excellence Award (PREA), in 2003 from the Province of Ontario, Canada. He served as the technical program committee chair/co-chair for the IEEE Globecom'16, the IEEE Infocom'14, IEEE VTC'10 Fall, IEEE Globecom'07, symposia chair for IEEE ICC'10, and chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the elected IEEE Communications Society vice president for Technical & Educational Activities, vice president for Publications, Member-at-Large on the Board of Governors, chair of the Distinguished Lecturer Selection Committee, member of IEEE Fellow Selection Committee. He was/is the editor-in-chief of the *IEEE Internet of Things Journal*, *IEEE Network*, *IET Communications*, and *Peer-to-Peer Networking and Applications*.