Query Integrity Assurance of Location-based Services Accessing Outsourced Spatial Databases

Wei-Shinn Ku[§] Ling Hu[†] Cyrus Shahabi[†] Haixun Wang[‡]
[§]Dept. of Computer Science and Software Engineering, Auburn University, USA
[†]Computer Science Department, University of Southern California, USA
[‡]IBM Thomas J. Watson Research Center, USA
weishinn@auburn.edu, {lingh,shahabi}@usc.edu, haixun@us.ibm.com

Abstract. Outsourcing data to third party data providers is becoming a common practice for data owners to avoid the cost of managing and maintaining databases. Meanwhile, due to the popularity of location-based-services (LBS), the need for spatial data (e.g., gazetteers, vector data) is increasing exponentially. Consequently, we are witnessing a new trend of outsourcing spatial datasets by data collectors. Two main challenges with outsourcing datasets is to keep the data private (from the data provider) and ensure the integrity of the query result (for the clients). Unfortunately, most of the techniques proposed for privacy and integrity do not extend to spatial data in a straightforward manner. Hence, recent studies proposed various techniques to support either privacy or integrity (but not both) on spatial datasets. In this paper, for the first time, we propose a technique that can ensure both privacy and integrity for outsourced spatial data. In particular, we first use a one-way spatial transformation method based on Hilbert curves, which encrypts the spatial data before outsourcing and hence ensures its privacy. Next, by probabilistically replicating a portion of the data and encrypting it with a different encryption key, we devise a technique for the client to audit the trustworthiness of the query results. We show the applicability of our approach for both k-nearest-neighbor and spatial range queries, the building blocks of any LBS application. Finally, we evaluate the validity and performance of our algorithms with real-world datasets.

1 Introduction

Due to the rapid advancements in network technology, the cost of transmitting a terabyte of data over long distances has decreased significantly in the past five years. In addition, the total cost of data management is five to ten times higher than the initial acquisition costs and it is likely that computing solution costs will be dominated by people costs in the future [22]. Consequently, there is a growing interest in outsourcing database management tasks to third parties that can provide these tasks for a much lower cost due to the economy of scale. This new outsourcing model has the apparent benefits of reducing the costs for running DBMSs independently and enabling enterprises to concentrate on their main businesses. On the other hand, there are two new concerns with this model. First, the data owner may not want to reveal the data to the data provider due

to either the sensitivity of the data (e.g., medical records) or the value of the data (e.g., Navteq road vector data). Second, data users need to be confident of the integrity of the data they receive. To illustrate, consider the scenario that Zagat (a data owner) gives its restaurants data to Google (a data provider) to make it available to its customers. First, Zagat does not want to reveal the data to Google as this is its main business and value-add. Second, a user asking for all restaurants with a certain rating wants to be confident that he is indeed receiving every and all the Zagat's restaurants and not some extra ones injected by Google (e.g., paid advertisers) or some missing ones deleted by Google. Several previous studies [1, 5] proposed solutions for supporting encrypted queries over encrypted databases to protect data owners' privacy. Another set of studies [14, 21, 17, 15, 3] focus on the problem of integrity in outsourced databases by guaranteeing that the results returned by the service provider for a client query are both *correct* and *complete*.

Meanwhile, due to the recent advances in wireless technology, mobile devices (e.g., cell phones, PDAs, laptops) with wireless communication capabilities are increasingly becoming popular. Hence, we are witnessing the emergence of many location-based services (LBS) that allow users to issue spatial queries from their mobile devices in a ubiquitous manner. Obviously, these applications are in desperate need of quality spatial data, resulting in an exponential increase in the customers of spatial data acquirers. Recent mergers between data providers (e.g., TomTom) and data owners (Tele Atlas) are the immediate consequences of this phenomenon. Therefore, the outsourcing of spatial data is becoming an appealing business model for both data owners and data providers. Unfortunately, while the exact same concerns of privacy and integrity exist for outsourcing the spatial data, there has not been much work in addressing these issues for spatial data except for [24, 25, 18]. To the best of our knowledge none of these studies consider both privacy and integrity at the same time. It is not clear whether the proposed approaches for one problem can easily extend to the other problem or even worse if they conflict with the solutions proposed to the other problem.

In this paper we propose an innovative approach that simultaneously ensures both the privacy and the integrity of outsourced spatial data. This is achieved by using space encryption as the basis of our approach and then devising techniques that enable the data users to audit the integrity of the query result for the most important spatial query types: range queries and k-nearest-neighbor queries (kNN). In particular, we first use a one-way spatial transformation method based on Hilbert curves, which encrypts the spatial data before outsourcing and hence ensures its privacy. Next, by probabilistically replicating a portion of the data and encrypting it with a different encryption key, we devise a technique for the client to audit the trustworthiness of the query results. We evaluated our approaches with both synthetic and real-world datasets. The process of computing Hilbert curves is efficient at client side when the space encryption key is known. Experiment results show that with more than 20% duplication of the original dataset on the server, clients can detect query result deletion attacks with very high confidence.

The remainder of this paper is organized as follows. Section 2 surveys the related work. The system architecture and an overview of our approach is introduced in Section 3. The design of our space encryption based data privacy protection approach is presented in Section 4. In Section 5, we address our spatial query integrity auditing solutions for both range query and k-nearest-neighbor query. The experimental validation of our design is presented in Section 6. Finally, Section 7 concludes the paper with a discussion of future work.

2 Related Work

The outsourcing of databases to a third-party service provider was first introduced by Hacigümüs et al. [6]. Generally, there are two security concerns in database outsourcing: *data privacy* and *query integrity*. We summarize the related researches as follows.

2.1 Data Privacy Protection

Hacigümüs et al. [5] proposed a method to execute SQL queries over encrypted databases. Their strategy is to process as much of a query as possible by the service providers, without having to decrypt the data. Decryption and the remainder of the query processing are performed at the client side. Agrawal et al. [1] proposed an order-preserving encryption scheme for numeric values that allows any comparison operation to be directly applied on encrypted data. Their technique is able to handle updates and new values can be added without requiring changes in the encryption of other values. Generally, existing methods enable direct execution of encrypted queries on encrypted datasets and allow users to ask identity queries over data of different encryptions. The ultimate goal of this research direction is to make queries in encrypted databases as efficient as possible while preventing adversaries from learning any useful knowledge about the data. However, researches in this field did not consider the problem of query integrity.

2.2 Query Integrity Assurance

In addition to data privacy, an important security concern in the database outsourcing paradigm is query integrity. Query integrity examines the trustworthiness of the hosting environment. When a client receives a query result from the service provider, it wants to be assured that the result is both *correct* and *complete*. Correct denotes that the query must be evaluated honestly with the outsourced database to retrieve the result and complete means that the result includes all the records satisfying the query. Devanbu et al. [3] proposed to employ the Merkle hash tree [12] to authenticate data records. The technique computes a signature based on the Merkle hash tree structure and distributes it to clients as a proof of correctness. Mykletun et al. [15] studied and compared several signature methods which can be applied in data authentication. The authors identified the problem of completeness, however they did not propose correspondent solutions. Pang et al. [17] utilized an aggregated signature to sign each

record with the information from neighboring records by assuming that all the records are sorted with a certain order. The method assures the completeness of a selection query by checking the aggregated signature. The *challenge token* idea was introduced in [21] for a server with outsourced databases to provide a proof of actual query execution which is then checked at the client side for integrity verification. Compared with [17], the mechanism supports more query types without assuming all the records are sorted. However, all the aforementioned solutions cannot support spatial queries directly.

For auditing spatial queries, Yang et al. [24] proposed the MR-tree which is an authenticated data structure suitable for verifying queries executed on outsourced spatial databases. The authors also designed a caching technique to reduce the information sent to the client for verification purposes. Four spatial transformation mechanisms are presented in [25] for protecting the privacy of outsourced private spatial data. The data owner selects transformation keys which are shared with trusted clients and it is infeasible to reconstruct the exact original data points from the transformed points without the key. Mouratidis et al. [14] proposed the *Partially Materialized Digest scheme* which avoids unnecessary query processing costs and outperforms existing solutions by employing separate indexes for the data and for their associated verification information. However, these researches [24, 25, 14] did not consider data privacy protection and query integrity auditing jointly in their design. The related work closest to ours is presented by Wang et al. [23] which focuses on numerical data integrity authentication. Nevertheless, the solution cannot be applied in auditing spatial queries because spatial locality information of records is destroyed after encryption.

3 System Overview

In this section, we introduce the architecture of our system and provide an overview of our approach.

3.1 System Architecture

Figure 1 illustrates the architecture of a spatial database outsourcing environment with three main components: mobile user, location-based service provider, and database owner. We consider mobile clients such as cell phones, personal digital assistants, and laptops, which are instrumented with Global Positioning System (GPS) receivers for continuous position information. Moreover, we assume that there are access points distributed in the system environment for mobile devices to communicate with LBS providers. Generally, mobile devices cannot store any significant amount of the outsourced data in local memory for integrity checking. Therefore, a feasible way of obtaining integrity assurance of query results for mobile devices is through asking queries and analyzing results. On the other hand, a LBS provider is able to store and access all the outsourced data for answering spatial queries from clients. However, LBS providers could be malicious (e.g., returning incomplete query results) and they are not trusted

by the clients. The third element – the database owner (e.g., possessing point of interest datasets) outsources its data management tasks to service providers (e.g., providing location-based services).

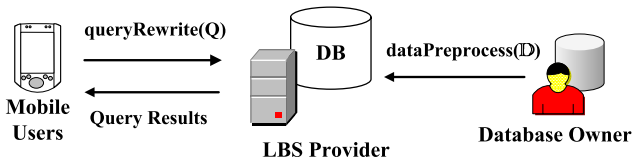

Fig. 1. The system architecture of spatial database outsourcing.

3.2 Overview of Our Approach

We assume that the database owner can embed additional information in the outsourced spatial dataset for query integrity verification. Let $\mathbb{D}$ denote the spatial database to be outsourced. The database owner first replicates a portion of $\mathbb{D}$ with randomly selected objects. Then, $\mathbb{D}$ and the replicated portion are encrypted with different Hilbert curve based encryption keys. Afterward, the two encrypted datasets are combined and stored at the LBS provider. We employ `dataPreprocess()` to denote the replication and encryption process and $\mathbb{D}_E =$ `dataPreprocess(`$\mathbb{D}$`)` to denote the spatial data stored at the service provider. For requesting LBS based on encrypted spatial databases, a mobile user rewrites spatial queries against $\mathbb{D}$ to spatial queries against $\mathbb{D}_E$ by making use of a query rewriting method `queryRewrite()`. In addition, the user also launches auditing queries for verifying a group of previously executed spatial queries. By exploiting the replicated data, the client is able to determine that the results are correct and complete and the confidence is beyond a user-specified level according to the replication ratio. The mechanism to discriminate a replicated data object from an original data object is only shared between the database owner and the users. LBS providers cannot tell the duplicated dataset from other encrypted data in the outsourced spatial database. Table 1 summarizes the set of notations of this paper.

4 Space Encryption based Privacy Protection

In this section, we first introduce the one-way function based space encryption solution. Next, space-filling curves are introduced and applied as one-way functions in our system for protecting the privacy of outsourced spatial data.

4.1 Space Encryption

In order to protect the privacy of outsourced spatial databases, we exploit the power of one-way functions to preserve privacy by encoding the locations of all

Symbol	Meaning
s	Spatial object
s_E	Encrypted spatial object
r	Data replication percentage
$\mathbb{D}$	Spatial database
$\mathbb{R}$	Query result set
$\|\mathbb{A}\|$	The number of elements in set $\mathbb{A}$
$V_{\mathcal{H}}$	Hilbert value
O	Order of a Hilbert curve
$\mathcal{T}$	One-way function for space encryption
I_d	Dual information
Ψ	Cryptographic signature
S_K	Symmetric key
SEK_P	Primary spatial encryption key
SEK_S	Secondary spatial encryption key
$Dist(p, q)$	The Euclidean distance between two objects p and q

Table 1. Symbolic notations.

spatial objects. A one-way function is easy to compute but difficult to invert, meaning that some algorithms can compute the function in polynomial time while no probabilistic polynomial-time algorithm can compute an inverse image of the function with better than negligible probability. Our space transformation method is capable to map each point from the original space to a point in the encrypted space to prevent the service provider from obtaining the original spatial object locations. Because we focus on managing spatial data, an ideal one-way transformation should respect the spatial proximity of the original space. If the encrypted space is able to maintain the distance properties of the original space, it will enable efficient evaluation of spatial queries. Transforming spatial object locations with such a locality-preserving one-way mapping can be viewed as encrypting the elements of the two-dimensional (2-D) space for securing privacy and facilitating spatial query processing. In this research, we apply the parameters of our space encryption function as the trapdoor [4] which is only provided to users to encode queries and decode the encrypted query results for retrieving the original spatial object positions.

4.2 Space Filling Curves

A space-filling curve is a continuous curve, which passes through every point of a closed space. The formal definition of a space-filling curve is as follows. If a mapping $f : I \rightarrow \mathbf{E}^n$ $(n \geq 2)$ is continuous, and $f(I)$, the image of I under f, has positive Jordan content (area for $n = 2$ and volume for $n = 3$), then $f(I)$ is called a space-filling curve. $\mathbf{E}^n$ denotes an n-dimensional Euclidean space. An important property of space-filling curves is that they retain the proximity and neighboring aspects of the indexed data. Because space-filling curves can preserve the locality between objects in the multidimensional space in the transformed linear space, we investigate the applicability of space-filling curves as ciphers for

preserving privacy of outsourced spatial databases. Since the main goal of this research is to provide both privacy protection and integrity assurance of location-based services with outsourced spatia databases, we focus on the transformation of a 2-D space which covers the locations of POIs. However, our solution can be easily extended to high dimensional space.

The Hilbert curve [7, 2] is a continuous fractal space-filling curve which is broadly used in multidimensional data management. The superior distance preserving properties [11] makes the Hilbert curve an ideal choice as a space cipher. In addition, the Hilbert curve achieves better clustering than the Z curve [16] and the Gray-coded curve [8]. Therefore, we apply the Hilbert curve in our system for encrypting the original space. As Ref [13], we define $\mathcal{H}_O^D$ for $O \geq 1$ and $D \geq 2$, as the O^{th} order Hilbert curve for a D-dimensional space. Consequently, $\mathcal{H}_O^D$ maps an integer set $[0, 2^{OD} - 1]$ into a D-dimensional integer space $[0, 2^O - 1]^D$. The mapping determines the Hilbert value $V_{\mathcal{H}}$ of each point in the original space based on their coordinates where $V_{\mathcal{H}} \in [0, 2^{OD} - 1]$. Accordingly, we can formulate the relationship in a two-dimensional space as $V_{\mathcal{H}} = \mathcal{T}(x, y)$ where x and y are the coordinate of a point in the original space and $\mathcal{T}$ is the one-way transformation function. Note that it is possible for two or more points to have the same Hilbert value in a given curve. Figure 2 illustrates an example of mapping 2-D space POIs into their Hilbert values. In the illustration, we can retrieve the Hilbert values of the points of interest A, B, C, and D as 0, 2, 8, and 12 respectively with an order two Hilbert curve. Depending on the desired resolution, more fine-grained curves can be recursively generated based on the Hilbert curve production rules.

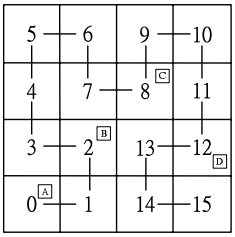

Fig. 2. The Hilbert curve transforms a 2-D space into corresponding Hilbert values.

Based on the aforementioned properties of the Hilbert curve, it can be employed as a one-way function to support space encryption. The curve parameters including the curve's starting point (x_0, y_0), curve order O, and curve orientation θ make up the *Space Encryption Key* (SEK) of the Hilbert curve based one-way function [9]. Consequently, adversaries who do not have the decryption key have to exhaustively check for all possible combinations of curve parameters to decipher the physical locations of interested objects. However, with reasonable curve parameters it is computationally impossible to reverse the transformation and retrieve the physical locations of interested points in polynomial time.

5 Spatial Query Integrity Auditing with Dual Space Encryption Keys

5.1 Dual Space Encryption

In order to audit the integrity of query results retrieved from outsourced spatial databases, we encrypt the original spatial database $\mathbb{D}$ with dual space encryption keys. We first encrypt $\mathbb{D}$ with a primary space encryption key SEK_P. Then, we replicate r percent of $\mathbb{D}$ and encode the duplicate with a secondary space encryption key SEK_S which possesses different curve parameters. Afterward, we combine the two encrypted datasets as $\mathbb{D}_E$ and store $\mathbb{D}_E$ at the service provider. After space encryption, a service provider can only see the Hilbert value of each spatial data object in $\mathbb{D}_E$ instead of their original coordinates. Since $\mathcal{T}$ is a one-way function, for any spatial object s in $\mathbb{D}_E$, a service provider cannot tell whether s was encoded by SEK_P or SEK_S. In addition, the Hilbert values generated by the two space encryption keys may overlap which makes it even more difficult to distinguish if an object is the original or the duplicate.

On the other hand, we need corresponding techniques to enforce query integrity on the client side. For any spatial object s in the query result set, a client should be able to verify whether s is a valid record of $\mathbb{D}$ and if s has a counterpart which is encrypted with another SEK. For supporting object verification, we encrypt the coordinate, non-spatial attributes, and dual information I_d with a symmetric key S_K which is shared by the database owner and all the clients. In addition, we apply cryptographic hash functions [20] to generate a signature Ψ for each spatial object with the coordinate and non-spatial attributes as the input message. The purpose of the dual information field is for clients to tell if a spatial object has a duplicate in the outsourced database. I_d has three values which stand for (i) primary encryption without duplication, (ii) primary encryption with duplication, and (iii) secondary encryption respectively. The structure of an encrypted spatial object stored in $\mathbb{D}_E$ is as follows:

$$s_E = \{V_{\mathcal{H}}, \{x, y, \textit{non-spatial attributes}, I_d\}_{S_K}, \Psi\}$$

For each server returned spatial object s_E, a client first decrypts s_E with the symmetric key and executes the cryptographic hash function for verifying the object with its attached signature. Since it is computationally infeasible to forge a cryptographic hash function generated signature, any tampering with the object will be detected. If the spatial object is valid, the client will check its I_d field for determining whether the object is an original or a duplicate. Replicated objects are utilized to audit query result integrity as described in the following two subsections.

5.2 Range Query

With a given range query Q_R, a client first identifies the Hilbert values covered by the range query based on the parameters of SEK_P. Afterward, the client queries

Algorithm 1 Query Integrity Assured Range Query (Q_R)

1: Compute Hilbert curve segments covered by Q_R on the curve defined by SEK_P and store the segments in $\mathbb{S}$
2: **for** each segment $e \in \mathbb{S}$ **do**
3: Retrieve the spatial objects covered by e and store them in $\mathbb{R}$
4: **end for**
5: Filter out the spatial objects encrypted with SEK_S in $\mathbb{R}$
6: **for** each spatial object $s \in \mathbb{R}$ **do**
7: Verify s with its signature
8: **if** s is a valid object and s has a duplicate **then**
9: Store s in $\mathbb{C}$
10: **else**
11: Report the anomaly to the client and exit
12: **end if**
13: **end for**
14: Create an auditing query Q_A based on Q_R and SEK_S
15: Compute Hilbert curve segments covered by Q_A on the curve defined by SEK_S and store the segments in $\mathbb{S}$
16: **for** each segment $e \in \mathbb{S}$ **do**
17: Retrieve the spatial objects covered by e and store them in $\mathbb{R}'$
18: **end for**
19: Filter out the spatial objects encrypted with SEK_P in $\mathbb{R}'$
20: **if** $\mathbb{C} \neq \mathbb{R}'$ **then**
21: Report the anomaly to the client and exit
22: **end if**
23: Return $\mathbb{R}$

the service provider for retrieving the objects covered by the query range. In order to hide the SEK parameters from malicious service providers, the client may interleave the Hilbert value segments covered by a group of range queries. After receiving the query result set $\mathbb{R}$, the client first filters out objects encrypted with SEK_S and verifies the validity of all the remaining objects with their attached signatures. If all the objects in $\mathbb{R}$ are valid, the client generates an auditing range query Q_A with the same query range size as Q_R and the parameters of SEK_S. If the service provider carries out queries honestly, the query result set of the auditing query must contain counterparts of all the objects with duplicates in $\mathbb{R}$. In practice, the client can launch a single auditing query for verifying a number of regular queries by combining their query ranges for saving resources.

Figure 3 demonstrates an example of range query integrity auditing. The query window of a range query Q_R covers three Hilbert curve segments, $[17-18]$, $[23-24]$, and $[27-31]$, based on the primary encryption key as shown in Figure 3(a). After receiving the three curve segments, the service provider retrieves all the spatial objects whose Hilbert values are embraced by the three curve sections and returns the retrieved spatial object set $\mathbb{R}$ as the query result. Then, the client removes records encrypted with SEK_S in $\mathbb{R}$, verifies the remaining records, and identifies the records which have duplicates by checking the I_d filed. Subse-

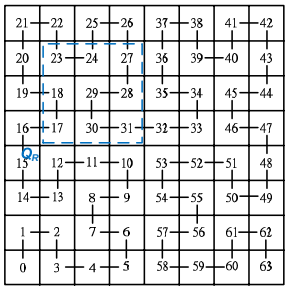

Fig. 3(a) Original range query Q_R.

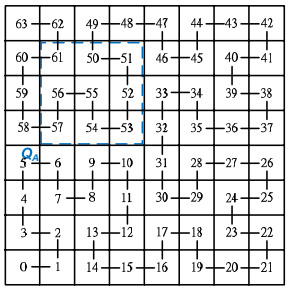

Fig. 3(b) Auditing range query Q_A.

Fig. 3. A range query Q_R covers three Hilbert curve segments based on SEK_P as illustrated in (a). The auditing query Q_A encloses two Hilbert curve segments based on SEK_S as demonstrated in (b).

quently, the client creates an auditing query Q_A with equal query range as Q_R on the Hilbert curve defined by SEK_S. In this example, Q_A encompasses two Hilbert curve segments, $[50-57]$ and $[61]$, based on the secondary encryption key. With the I_d field, the client is able to filter out the objects which are encrypted with SEK_P in the result of Q_A. Finally, the client checks if all the duplicates retrieved by Q_A have counterparts in $\mathbb{R}$. If there is any mismatch, the discrepancy proves that the service provider is malicious. The complete procedure of a *Query Integrity Assured Range Query* (QIARQ) is formalized in Algorithm 1.

5.3 k Nearest Neighbor Query

We design a *Query Integrity Assured k Nearest Neighbor* (QIAKNN) search algorithm by extending our range query solution in Section 5.2. For a given kNN query point Q located at position (x_Q, y_Q), a client first employs SEK_P to compute $V_{\mathcal{H}} = \mathcal{T}(x_Q, y_Q)$ as the query point in the encrypted space. Because there is r percent duplicate data in $\mathbb{D}_E$ which should be filter out from query results, we multiply k by $(1 + r)$ to get k' and apply k' as the query parameter. Thereafter, the client transmits the values of $V_{\mathcal{H}}$ and k' to the service provider for retrieving k' nearest neighbors of Q. The service provider searches $\mathbb{D}_E$ with both directions (ascending and descending) of $V_{\mathcal{H}}$ until k' closest spatial objects are found and then returns the query result set $\mathbb{R}$ to the client. After being receipt of $\mathbb{R}$, the client first removes objects encrypted with SEK_S and checks if there are k objects leftover in $\mathbb{R}$. If $\mathbb{R}$ contains fewer than k objects, the client repeats the aforementioned steps with a multiple of r until obtaining k valid objects. Subsequently, the client retrieves the object s^* which has the longest distance to Q in $\mathbb{R}$. Because of loss of a dimension in the encrypted space, the objects in $\mathbb{R}$ may not precisely match the actual k nearest neighbors of Q. Consequently, the client utilizes the distance between Q and s^* ($Dist(Q, s^*)$) as a *search upper bound* and launches a range query Q_R with $Dist(Q, s^*)$ to decide the query window size. Following acquiring $\mathbb{R}'$ as the result of Q_R, the client audits the range query result as described in Section 5.2. Because $Dist(Q, s^*)$ is the search

Algorithm 2 Query Integrity Assured k Nearest Neighbor Query(Q, k)

```
1: Compute V_H = T(x_Q, y_Q) based on SEK_P
2: Set δ = true, λ = 1, and γ = 0
3: while δ do
4:    Set R = ∅
5:    k' = k(1 + λ * r)
6:    R ∪ Retrieve k' objects closest to Hilbert value V_H from the server
7:    Filter out the spatial objects encrypted with SEK_S in R
8:    if |R| ≥ k then
9:       δ = false
10:   else
11:      λ = λ + 1
12:   end if
13: end while
14: for i = 0; i < |R|; i++ do
15:   if Dist(Q, s_i) > γ then
16:      γ = Dist(Q, s_i)
         /* s_i ∈ R */
17:      s* = s_i
18:   end if
19: end for
20: Compute the edge length of Q_R by Dist(Q, s*)
21: R' = QIARQ(Q_R)
22: Set γ = ∞, λ = 0, and R = ∅
23: for j = 0; j < |R'|; j++ do
24:   if λ < k then
25:      λ = λ + 1
26:      R = R ∪ s_j
27:   else
28:      sort R in ascending order of distance to Q and retrieve the last element as s_k
29:      if Dist(Q, s_k) > Dist(Q, s_j) then
30:         Replace s_k with s_j
31:      end if
32:   end if
33: end for
```

upper bound, the client has to identify the top k objects in $\mathbb{R}'$ based on their distance to Q to acquire the final query result.

We illustrate k-nearest-neighbor query integrity auditing with an example in Figure 4. The client first encodes the location of the query point with SEK_P and computes its $V_{\mathcal{H}} = 30$. Afterward, the client launches a kNN query with the numbers of $V_{\mathcal{H}}$ and k'. The service provider searches $\mathbb{D}_E$ for objects with Hilbert values ≥ 30 and < 30 in parallel until k' objects are found as demonstrated in Figure 4(a). Next, the client interacts with the server until k objects encrypted in SEK_P are retrieved. Among the k valid objects, assume the one which has the longest distance to Q has Hilbert value 32 and then we can obtain the search window edge length as five units. Subsequently, the client launches a

query integrity assured range query for searching k nearest objects of Q as the exact query result as showed in Figure 4(b).

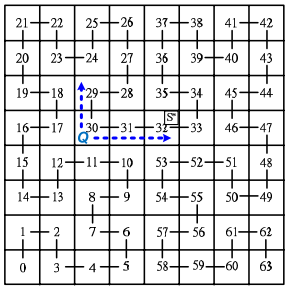

Fig. 4(a) Finding the search upper bound.

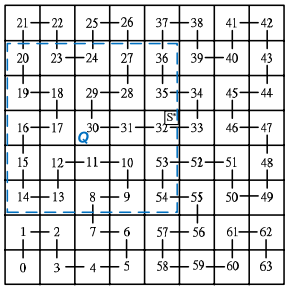

Fig. 4(b) Auditing range query Q_A.

Fig. 4. A query integrity assured k-nearest-neighbor query.

5.4 Attack-aware Auditing Query Composition

The purpose of our dual space encryption design is to allow for sophisticated cross examination. Mobile users carry out cross examination against a single spatial database that has two different encryptions. However, negligent auditing queries launched by clients may reveal critical information to allow malicious LBS providers to detect the correspondence among the data with different encryption keys. For example, assume a client launches an auditing query after every regular query. Then, a malicious service provider can easily learn the relationship between the two queries and remove the query results of both queries to jeopardize future queries without being detected by clients.

In order to defend against the aforementioned attack, we need more advanced solutions for composing auditing queries. Generally, we want to create a checking query Q_A, which will not leak any correspondence information among the data objects in $\mathbb{D}_E$. In addition, Q_A should be hard to differentiate from other regular queries. Consequently, the main principle is to apply a single query to evaluate the integrity of multiple queries. Because spatial queries launched by the same mobile client usually exhibit locality [10], the query range overlap between successive queries from identical user is significant. By evaluating the integrity of multiple queries at a time, we can improve security and decrease integrity auditing overhead to save energy of mobile devices. Based on the memory capacity, a client can decide the threshold to generate a checking query for a group of executed regular queries $\mathcal{Q} = \{q_1, \ldots, q_n\}$ by merging their query regions. Only queries whose results contain replicas should be included in $\mathcal{Q}$.

6 Experimental Validation

We use Hilbert curves as a space encoding approach to encrypt spatial information in outsourced databases. It has been proved in [9] that without knowing

the space encryption key, a brute force attack will need to exhaustively search all possible key combinations and the complexity of the attack is $O(2^{4b})$ where b is the number of bits for each parameter. Hence, the Hilbert curve based encryption method is employed as a one way function in our design. Table 2 illustrates two synthetic datasets and three real-world datasets utilized in our experiments. The two synthetic datasets of 10K data points each represent uniform and skew distributions, respectively. Los Angeles is a dataset containing around 10K restaurants inside a geographic area measuring 26 miles by 26 miles in the City of Los Angeles, California. The last two datasets consist of points of interest (POI) across California (61K) and North America (556K). Our query integrity assurance algorithms were implemented in Java and the experiments were conducted on a Windows Vista PC with Intel Core 2 Duo 3.16GHz processor and 4GB memory. All simulation results were recorded after the system model reached a steady state.

Name	Number of POIs	Source
Uniform	10,163	Synthetic
Skewed	10,163	Synthetic
Los Angeles (LA)	10,163	NAVTEQ
California (CA)	62,556	US Census Bureau
North America (NA)	569,120	US Census Bureau

Table 2. The simulation datasets.

6.1 Encoded POI Density

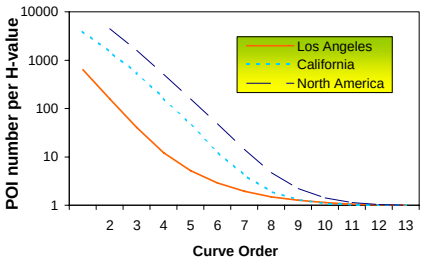

Fig. 5(a) POI density on different size of datasets.

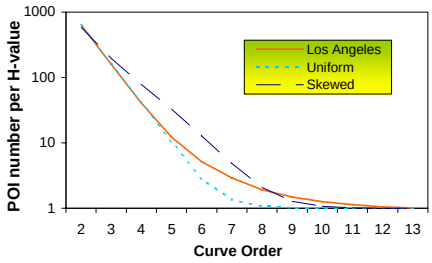

Fig. 5(b) POI density on different data distributions.

Fig. 5. The relationship between curve order and POI density.

We first show the relationship between Hilbert curve orders and the number of POIs encoded by one Hilbert value (POI density). A higher Hilbert curve order increases the security level of the corresponding space encryption key while on the other hand, a higher curve order will incur higher computational complexity. As Figure 5 shows with both synthetic and real-world datasets the number of POIs

per Hilbert value decreases rapidly as the curve order increases. The average number of POIs per non-empty Hilbert value reaches 1 when the curve order is greater than 12. An empty Hilbert value means that there is no POI associated with it and we discard these empty values during the initialization process. Hence, we use the default curve order of 12 for space encryption unless specified explicitly.

6.2 Spatial Database Outsourcing Initialization

There are three major operations in the initialization process for our spatial database outsourcing approach: (1) computing Hilbert values for all data objects based on their locations; (2) encrypting each data object with the symmetric key; and (3) calculating the cryptographic signature for each object. There are various algorithms for (2) and (3) and the cost of each may vary. We used the Blowfish encryption algorithm [19] and MD5 (Message-Digest algorithm 5) for signature computation. We perform the data initialization process on all the three real-world datasets with 40% duplication rate and curve order of 12. The cost of each operation in Figure 6 shows that computing the Hilbert values is efficient and less expensive than the other two operations.

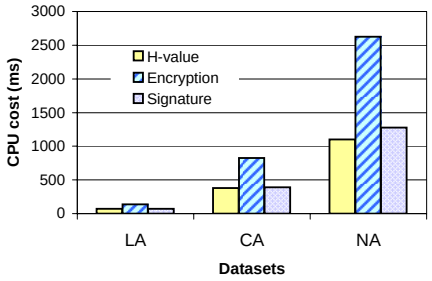

Fig. 6. Initialization cost of the proposed spatial database outsourcing approach.

6.3 Query Processing on the Client Side

For range queries, the client performs a transformation from range query window to Hilbert curve segments and sends these segments to the service provider. A kNN query, as described in Algorithm 2, is split into two parts: retrieve k nearest POIs simply based on the Hilbert value of the query point and launch a range query using the distance of the k^{th} point in the previous operation as the search upper bound.

The transformation cost on the client side is analyzed in this experiment. We extended the size of range query windows from 0.01 to 0.05 on the normalized dataset and varied curve orders from 10 to 15. Figure 7 demonstrates the average cost of 50 range queries on the Los Angeles dataset with the aforementioned

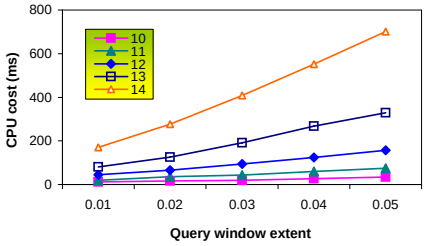

Fig. 7. Query processing cost on the client side.

settings. Based on the results, we can see that it is efficient to compute Hilbert curve segments with a given SEK and the client does not need to store any information other than the SEK. Hilbert curve segments are represented by the start and the end values and are transmitted to the service provider to retrieve spatial objects.

6.4 Integrity Auditing

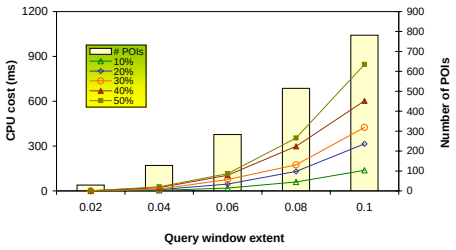

Fig. 8. Cost of authenticating query results.

Clients employ QIARQ and QIAKNN to verify if the spatial query results are both correct and complete when receiving query results. There are three operations in the query result authentication process: (1) decrypt each data object; (2) verify the signature of every retrieved object; and (3) check the counterpart existence of each object with a duplicate. The cost of (1) and (2) are constant per object given the same encryption method. Therefore, we only show the cost of (3) in Figure 8 for range queries at different range extents and data replication percentage. Verifying query results is efficient when the size of the data returned is small, which is the case in general for common kNN queries and range queries with small extents. To better illustrate the authentication cost, we use the California dataset here with query window extent varying from 0.02 to 0.1 and replication percentage changing from 10% to 50% on the normalized data. The returned POI number and CPU cost increase linearly when we enlarge the query window and replication ratio as shown in Figure 8.

6.5 Communication Cost

We study the communication cost by considering the size of data transferred between the client and the service provider. Network delays and packet retransmission due to unstable connections are not the focus of this research and hence they are not considered in this experiment. A round trip of a query-and-answer process between a client and a server can be split into two parts and each of them are affected by different factors. The query transfer from a client to a server is composed of multiple Hilbert curve segments and the size of which is related to the Hilbert curve order in the applied SEK. The result returning back from the service provider contains all the POIs with respect to the query and the size is determined by the distribution of POIs, the extent of the query range (or the number k for a kNN query), and the replication percentage of the outsourced database. We assume the size of every data object is 1KB. In Figure 9(a), the communication cost is measured by the number of segments transmitted and in Figure 9(b), it is measured by the number of objects in the transmission. The experiments were performed on the Los Angeles dataset and the trend was similar for other datasets. As it can be observed from the figures, the communication cost increases linearly when we expand the curve order or the replication percentage with the same query window extent.

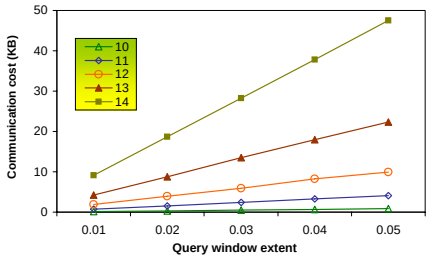

Fig. 9(a) Client-to-server communication cost.

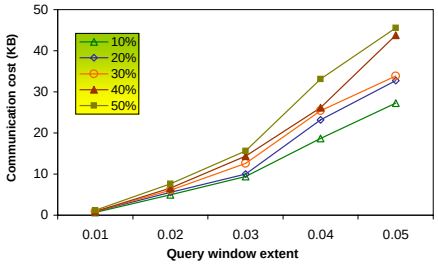

Fig. 9(b) Server-to-client communication cost.

Fig. 9. The cost of communication between a client and a service provider.

6.6 Against Malicious Attacks

Modifying and adding data objects in an outsourced spatial database can be easily detected by our integrity auditing algorithms, which results in one of the two cases: unable to perform decryption on the tampered data or inconsistent cryptographic signatures. Consequently, the attack model studied in this experiment primarily focuses on data object deletion by malicious service providers. We conducted the simulation on the Los Angeles dataset using randomly generated queries with the extent of 0.04 on the normalized coordinates. With different data replication ratio, the server launches random deletion attacks on query results. Figure 10 shows the probability that the attacker can escape from client

auditing process versus the total number of data objects deleted from a query result. As we can see from the figure, with more than 20% replication, the probability of not detecting a deletion declines rapidly as the service provider deletes more data objects.

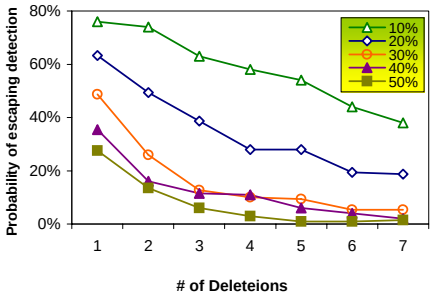

Fig. 10. Probability of escaping detection of deletion attacks.

7 Conclusions

Outsourcing of spatial databases for supporting location-based services has become a trend in recent years due to the economy of scale. Existing solutions are designed for data privacy protection or query integrity auditing, respectively, instead of considering both data privacy and query integrity as a whole. We have introduced query integrity assured algorithms for both range query and k-nearest-neighbor query with space encryption techniques to secure data privacy. We have demonstrated through simulation results that our mechanisms have remarkable performance. For future work, we plan to extend our algorithms to support more spatial query types such as spatial join, spatial path queries, etc.

Acknowledgements

This research has been funded in part by the US National Science Foundation (NSF) grants IIS-0238560 (PECASE), IIS-0534761, IIS-0742811, CNS-0831502 (CT), and CNS-0831505 (CT), and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-Preserving Encryption for Numeric Data. In *SIGMOD Conference*, pages 563–574, 2004.
2. A. R. Butz. Alternative Algorithm for Hilbert's Space-Filling Curve. *IEEE Trans. Comput.*, 20(4), 1971.

3. P. T. Devanbu, M. Gertz, C. U. Martel, and S. G. Stubblebine. Authentic Third-party Data Publication. In *DBSec*, pages 101–112, 2000.
4. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
5. H. Hacigümüs, B. R. Iyer, C. Li, and S. Mehrotra. Executing SQL over Encrypted Data in the Database-service-provider Model. In *SIGMOD Conference*, pages 216–227, 2002.
6. H. Hacigümüs, S. Mehrotra, and B. R. Iyer. Providing Database as a Service. In *ICDE*, page 29, 2002.
7. D. Hilbert. Ueber die stetige Abbildung einer Linie auf ein Flchenstck. *Mathematische Annalen*, (38):459–460, 1891.
8. H. V. Jagadish. Linear Clustering of Objects with Multiple Atributes. In *SIGMOD Conference*, pages 332–342, 1990.
9. A. Khoshgozaran and C. Shahabi. Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy. In *SSTD*, pages 239–257, 2007.
10. W.-S. Ku, R. Zimmermann, and H. Wang. Location-Based Spatial Query Processing in Wireless Broadcast Environments. *IEEE Trans. Mob. Comput.*, 7(6):778–791, 2008.
11. J. K. Lawder and P. J. H. King. Querying multi-dimensional data indexed using the hilbert space-filling curve. *SIGMOD Record*, 30(1):19–24, 2001.
12. R. C. Merkle. A Certified Digital Signature. In *CRYPTO*, pages 218–238, 1989.
13. 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.*, 13(1):124–141, 2001.
14. K. Mouratidis, D. Sacharidis, and H. Pang. Partially Materialized Digest Scheme: An Efficient Verification Method for Outsourced Databases. *VLDB J.*, 18(1):363–381, 2009.
15. E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and Integrity in Outsourced Databases. In *NDSS*, 2004.
16. J. A. Orenstein. Spatial Query Processing in an Object-Oriented Database System. In *SIGMOD Conference*, pages 326–336, 1986.
17. H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan. Verifying Completeness of Relational Query Results in Data Publishing. In *SIGMOD Conference*, pages 407–418, 2005.
18. S. Papadopoulos, D. Papadias, W. Cheng, and K.-L. Tan. Separating Authentication from Query Execution in Outsourced Databases. In *ICDE*, 2009.
19. B. Schneier. Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In *Fast Software Encryption, Cambridge Security Workshop*, pages 191–204, London, UK, 1994. Springer-Verlag.
20. B. Schneier. *Applied Cryptography (2nd ed.): Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1996.
21. R. Sion. Query Execution Assurance for Outsourced Databases. In *VLDB*, pages 601–612, 2005.
22. I. Sommerville. *Software Engineering (8th Edition)*. Addison Wesley, 2006.
23. H. Wang, J. Yin, C.-S. Perng, and P. S. Yu. Dual Encryption for Query Integrity Assurance. In *CIKM*, pages 863–872, 2008.
24. Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios. Spatial Outsourcing for Location-based Services. In *ICDE*, pages 1082–1091, 2008.
25. M. L. Yiu, G. Ghinita, C. S. Jensen, and P. Kalnis. Outsourcing of Private Spatial Data for Search Services. In *ICDE*, 2009.