Geoinformatica (2013) 17:97–124
DOI 10.1007/s10707-012-0156-9

A query integrity assurance scheme for accessing outsourced spatial databases

Wei-Shinn Ku · Ling Hu ·
Cyrus Shahabi · Haixun Wang

Received: 28 November 2010 / Revised: 13 January 2012 /
Accepted: 26 February 2012 / Published online: 15 March 2012

Abstract With the trend of cloud computing, outsourcing databases to third party service providers is becoming a common practice for data owners to decrease the cost of managing and maintaining databases in-house. In conjunction, due to the popularity of location-based-services (LBS), the need for spatial data (e.g., gazetteers, vector data) is increasing dramatically. Consequently, there is a noticeably new tendency of outsourcing spatial datasets by data collectors. Two main challenges with outsourcing datasets are to keep the data private (from the data provider) and to 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

W.-S. Ku (✉)
Department of Computer Science and Software Engineering,
Auburn University, Auburn, AL 36849, USA
e-mail: weishinn@auburn.edu

L. Hu · C. Shahabi
Computer Science Department, University of Southern California,
Los Angeles, CA 90089, USA

L. Hu
e-mail: lingh@usc.edu

C. Shahabi
e-mail: shahabi@usc.edu

H. Wang
Microsoft Research Asia, Beijing Sigma Center, Beijing, China 100190
e-mail: haixunw@microsoft.com

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 queries and spatial range queries, which are the building blocks of any LBS application. We also design solutions to guarantee the freshness of outsourced spatial databases. Finally, we evaluate the validity and performance of our algorithms with security analyses and extensive simulations.

Keywords Outsourcing · Privacy · Query integrity · Spatial databases

1 Introduction

The cost of transmitting a terabyte of data over long distances has decreased significantly in the past five years due to the rapid advancements in network technology. In addition, the total cost of data management is five to ten times higher than the initial acquisition costs, and it is likely that people costs will dominate computing solution costs in the future [30]. Meanwhile, Cloud computing provides flexible resources that can easily scale up or down (based on user demand), effectively reducing the operational and maintenance expenses for data owners (DOs). Consequently, there is a growing interest in outsourcing database management tasks to third parties (Cloud service providers) who can provide these data services 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.

However, the new outsourcing model also introduced two major concerns. First, the data owner may not want to reveal the data to the service provider due to either the sensitivity of the data (e.g., medical records, call detail records, or military data) or the value of the data (e.g., Navteq road vector data). For example, one of the reasons for the failure of Google Health is that "Google Health is not trustworthy" [6]. And patients are not willing to share their medical data with companies like Google or Microsoft. Second, data users need to be confident that the data received from the third party provider are still trustworthy. In fact, as the service provider is not the real owner of the data, they might return dishonest results to query clients out of its own interests. Additionally, the query results might be tampered with by malicious attackers who could substitute real results with fake ones or delete results with higher rankings. Consequently, the query results may not be trustworthy.

To illustrate this, consider the scenario where Zagat Survey (a data owner) gives its restaurants' data to Google (a data provider) in order to make it available to Zagat's 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 above a certain rating within three miles wants to be confident that he is indeed receiving all the restaurants from Zagat that satisfy the query, but with no extra ones injected by Google (e.g., paid advertisers) or missing ones deleted by Google. Furthermore, a data provider could be compromised and provide malicious query results, which benefits the attacker (e.g., a competitor data provider). For example, in April 2011, Sony PlayStation Network was broken into by hackers, exposing the personal information of 77 million people around the world [3, 26]. In addition, in June 2011, Dropbox temporarily allowed visitors to log in using any of its 25 million users' accounts due to a software glitch [3, 7]. As a result, although service providers like Google would not provide false results to their customers in order not to harm their own quality of service, even for some immediate benefits (e.g., advertisement income), they still might act involuntarily and return false results, e.g., while under attack, or the results might be tampered with while transmitting query results to clients (e.g., network attack). Therefore, providing a mechanism that allows query clients to verify the integrity of query results is indeed necessary. Several previous studies [1, 9] proposed solutions for supporting encrypted queries over encrypted databases to protect data owners' privacy. Another set of studies [4, 20, 21, 23, 29] 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, tablets, netbooks) with wireless communication capabilities are increasing in popularity. 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. The immediate consequences of this phenomenon are the recent mergers between data providers (e.g., TomTom) and data owners (Tele Atlas). Consequently, 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 [20, 25, 33, 34]. 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 either problem could be easily extended to the other problem or, even worse, if they could conflict with each other.

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 firstly 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). This paper is based on our earlier paper [14] which did not provide freshness guarantees for outsourced spatial databases and lacked theoretical analysis of the proposed techniques. In order to achieve a complete query integrity assurance scheme, we design corresponding freshness auditing solutions for UPDATE, INSERT, and DELETE operations. In addition, we formally analyze the query integrity assurance properties of our solutions against attacks from malicious service providers. The contributions of our work are as follows:

- We employ a one-way spatial transformation method based on the Hilbert curve, which encrypts the spatial data before outsourcing and, hence, ensures its privacy.

- We devise techniques for the client to audit the trustworthiness of the query results by probabilistically replicating a portion of the outsourced data and encrypting it with a different encryption key.
- We design mechanisms to provide freshness guarantees for outsourced spatial databases.
- We formally analyze the integrity assurance properties of our solutions against attacks from malicious service providers.
- We evaluate our approaches with both synthetic and real-world datasets. Experiment results show that with more than a 20% duplication of the original dataset on the server, clients have a significantly higher probability of detecting query result deletion attacks.

The rest of this paper is organized as follows. Section 2 surveys the related work. We introduce the system architecture and an overview of our approach in Section 3. The design of our space encryption based data privacy protection approach is presented in Section 4. In Section 5, we introduce our spatial query integrity auditing solutions for both range queries and *k*-nearest-neighbor queries. We propose our solutions to guarantee the freshness of outsourced spatial databases in Section 6. In Section 7, we formally analyze the integrity assurance properties of our techniques against malicious attacks. The experimental validation of our design is presented in Section 8. Finally, Section 9 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. [10]. Generally, there are two security concerns in database outsourcing: *data privacy* and *query integrity*. We summarize related research as follows.

2.1 Data privacy protection

Hacigümüs et al. [9] 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, query integrity is not taken into account in this field and it is assumed that mobile users trust everything returned by a service provider.

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, they want to be assured that the result is *authentic*, *correct*, and *complete*. Authenticity ensures the users that returned results originally come from the data owner; 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. [4] proposed employing the Merkle hash tree [18] to authenticate data records. The technique computes a signature based on the Merkle hash tree structure and distributes it to clients as proof of correctness. Mykletun et al. [21] studied and compared several signature methods, which can also be applied in data authentication. The study identifies the problem of completeness; however, it does not propose correspondent solutions. Pang et al. [23] utilizes an aggregated signature in order to sign each record with the information from neighboring records by assuming that all the records are sorted in a certain order. The method assures the completeness of a selection query by checking the aggregated signature. The *challenge token* idea, introduced by Sion [29], is 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 [23], the mechanism supports more query types without assuming all the records are sorted. However, the aforementioned solutions do not support spatial queries directly.

For auditing spatial queries, Yang et al. [33] 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. In order to protect data privacy, four spatial transformation mechanisms are presented in [34, 35] for protecting the privacy of outsourced private spatial data. The data owner selects transformation keys which are shared with trusted clients. It is not feasible to reconstruct the exact original data points from the transformed points without the key. Mouratidis et al. [20] 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, research works in [20, 33, 34] did not jointly consider data privacy protection and query integrity auditing in their design. The related work closest to ours is presented by Wang et al. [31] which focuses on numerical data integrity authentication. Nevertheless, the solution cannot be applied to audit spatial queries because spatial locality information of records is destroyed after encryption.

2.3 Freshness guarantee of outsourced databases

For outsourced databases, freshness guarantee mechanisms assure that queries are executed against the latest data records, rather than out-of-date versions. The cost of freshness guarantee solutions is usually expensive since it takes significant resources to monitor whether or not an outsourced database is up-to-date. Xie et al. [32] proposed solutions for providing freshness guarantees to support outsourced databases. Their mechanism can be added to existing query integrity assurance schemes (both authenticated data structure-based and probabilistic-based approaches). Pang et al. [24] designed a protocol for outsourced dynamic databases that allows new data to be disseminated immediately, while ensuring that outdated values beyond a pre-set age can be detected. However, without considering the unique characteristics of spatial data, the techniques in [24, 32] cannot be employed directly to guarantee the freshness of outsourced spatial databases for supporting location-based services.

3 System overview

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

3.1 System architecture

Figure 1 illustrates the architecture of a spatial database outsourcing environment with four main components: mobile user, trusted gateway, location-based service provider, and database owner. We consider mobile clients, such as cell phones, tablets, and netbooks, instrumented with Global Positioning System (GPS) receivers for continuous position information. The trusted gateway is a stand-alone server between mobile users and the LBS provider. The purpose of the trusted gateway is to coordinate spatial queries from clients and to rewrite client queries with space encryption keys. In addition, the trusted gateway composes auditing queries for assuring integrity of previously executed spatial queries and freshness of the outsourced spatial database. Afterward, the trusted gateway forwards the rewritten queries to the LBS provider. With database servers, the 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 or injecting fake data records) and they are not trusted by the clients. The last element – the database owner (e.g., possessing point of interest datasets) outsources their data management tasks to service providers (e.g., providing location-based services). We denote the joint component of mobile user and trusted gateway as *client* hereafter in this paper.

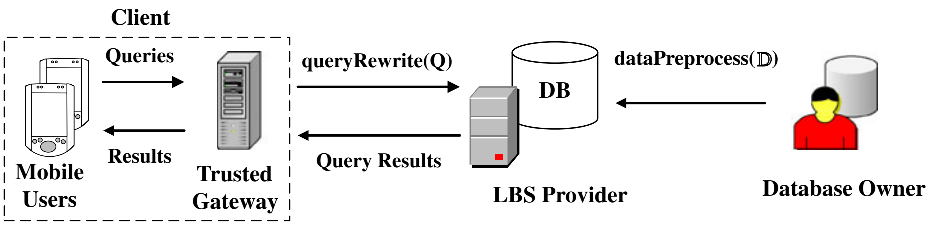


Fig. 1 The system architecture of spatial database outsourcing

3.2 Overview of our approach

We assume that the database owner D_o 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, the client 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 client also launches auditing queries for verifying a group of previously executed spatial queries. By verifying signatures and exploiting the replicated data, the client is able to determine that the results are authentic, correct, and complete. The knowledge of how to differentiate a replicated data object from an original data object is only shared between the database owner and the trusted gateway. LBS providers cannot tell the duplicated dataset from other encrypted data in the outsourced spatial database. Also, the client and the database owner audit the freshness of the outsourced spatial database collaboratively through fake database update operations. Table 1 summarizes the set of notations of this paper.

Table 1 Symbolic notations

Symbol	Meaning
S	Spatial object
S_e	Encrypted spatial object
r	Data replication percentage
t_i	Fake operation time slot
u	The total amount of time a fake record stays in the outsourced DB
D_o	Database owner
$\mathbb{D}$	Spatial database
$\mathbb{D}_e$	Encrypted spatial database
$\mathbb{R}$	Query result set
$\lvert\mathbb{A}\rvert$	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
T_s	Update time stamp
U_r	Update region of a fake update operation
Ψ	Cryptographic signature
S_k	Symmetric key
SEK_P	Primary spatial encryption key
SEK_S	Secondary spatial encryption key
K_P	Private key of D_o
$Dist(p, q)$	The Euclidean distance between two objects p and q

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 for protecting the privacy of outsourced spatial data in our system.

4.1 Space encryption

We exploit the power of one-way functions to preserve privacy by encoding the locations of all spatial objects in order to protect the privacy of outsourced spatial databases. 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 of mapping each point from the original space to a point in the encrypted space in order 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 could 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 [5], which is only provided to the trusted gateway, 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 partition 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 a 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 spatial databases, we focus on the transformation of a 2-D space that covers the locations of POIs. However, our solution can also be easily extended to high dimensional spaces.

The Hilbert curve [2, 11] is a continuous fractal space-filling curve which is broadly used in multidimensional data management. The superior distance preserving properties [17] makes the Hilbert curve an ideal choice as a space cipher. In addition, the Hilbert curve achieves better clustering than the Z curve [22] and the Gray-coded curve [12]. Therefore, we apply the Hilbert curve in our system for encrypting the original space. As in [19], we define $\mathcal{H}_O^D$ for $O \geq 1$ and $D \geq 2$, as

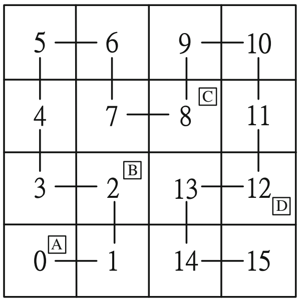


Fig. 2 The Hilbert curve transforms a 2-D space into corresponding Hilbert values

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 coordinates of a point in the original space, and $\mathcal{T}$ is the one-way transformation function. 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. Note that it is possible for two or more points to have the same Hilbert value in a given curve.

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 [13]. Consequently, adversaries who do not have the decryption key have to exhaustively check for all possible combinations of curve parameters in order 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

We encrypt the original spatial database $\mathbb{D}$ with dual space encryption keys in order to audit the integrity of query results retrieved from outsourced spatial databases. We first encrypt $\mathbb{D}$ with a primary space encryption key SEK_P. Then, we replicate r percent of $\mathbb{D}$ and encode the duplicates 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 then store $\mathbb{D}_e$ on 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 if 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 coordinates, non-spatial attributes, dual information I_d, and time stamp T_s with a symmetric key S_k, which is shared by the database owner and the trusted gateway. In addition, we apply cryptographic hash functions [28] to generate a signature Ψ for each spatial object with the coordinates, non-spatial attributes, I_d, and T_s as the input message. Afterward Ψ is encrypted with the private key of D_o. The purpose of the dual information field is for the client 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:

$$S_e = \left\{V_{\mathcal{H}}, \{x, y, \textit{non-spatial attributes}, I_d, T_s\}_{S_k}, \{\Psi\}_{K_p}\right\}$$

For each server returned spatial object S_e, the client first decrypts S_e with the symmetric key and executes the cryptographic hash function for verifying the object with its attached signature, which is decrypted with D_o's public key. Any tampering with the object will be detected, since it is computationally infeasible to forge a cryptographic hash function generated signature. If the spatial object is valid, the client will check its I_d field to determine 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, the client first identifies the Hilbert values covered by the range query based on the parameters of SEK_P. Afterward, the client queries the service provider for retrieving the objects covered by the query range. 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 from mobile users 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 Fig. 3a. After receiving the three curve segments, the service provider retrieves all the spatial objects whose Hilbert values are included by the three curve sections, and then returns the retrieved spatial object set $\mathbb{R}$ as the query result. Then, the client removes

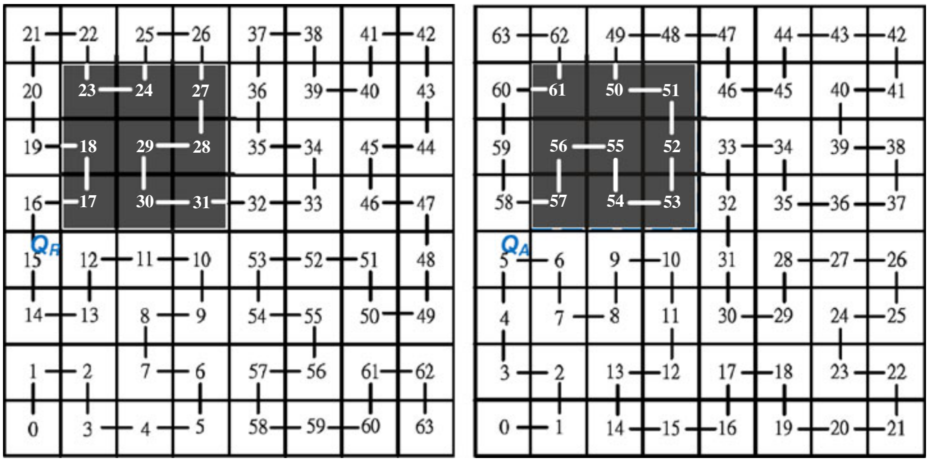

(a) Original range query Q_R. (b) Auditing range query Q_A.

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**)

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. Subsequently, 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 to see 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

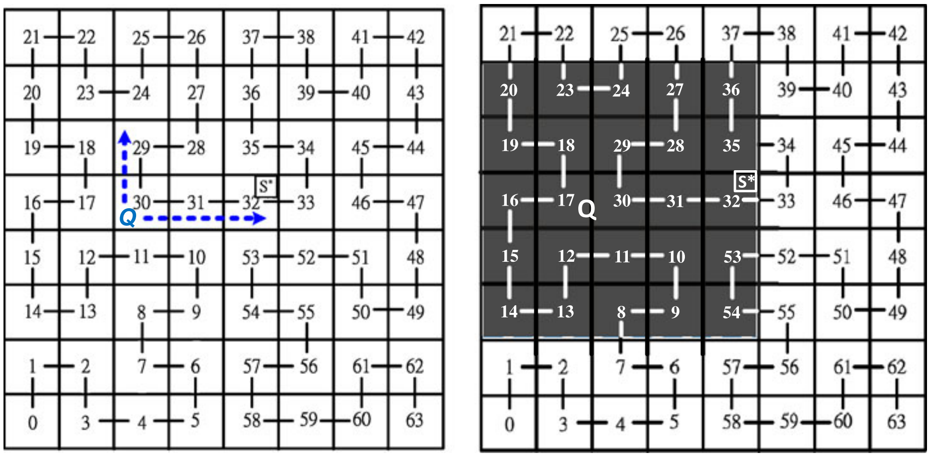
We design a *Query Integrity Assured k Nearest Neighbor* (QIAKNN) search algorithm by extending our range query solution in Section 5.2. For a given kNN query point Q located at position (x_Q, y_Q), a client first employs SEK_P to compute $V_{\mathcal{H}} = \mathcal{T}(x_Q, y_Q)$ as the query point in the encrypted space. Because there is r percent duplicate data in $\mathbb{D}_e$, which should be filtered 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 in order to retrieve k' nearest neighbors of Q. The service provider searches $\mathbb{D}_e$ in 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. On the receiving of $\mathbb{R}$, the client first removes objects encrypted with SEK_S, and then checks to see 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 the loss

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 $S$ has an invalid signature then
9:         Report the anomaly to the client and exit
10:    end if
11:    if $S$ has a duplicate then
12:        Store $S$ in $\mathbb{C}$
13:    end if
14: end for
15: Create an auditing query $Q_A$ based on $Q_R$ and $SEK_S$
16: Compute Hilbert curve segments covered by $Q_A$ on the curve defined by $SEK_S$ and store the segments in $\mathbb{S}$
17: for each segment $e \in \mathbb{S}$ do
18:     Retrieve the spatial objects covered by $e$ and store them in $\mathbb{R}'$
19: end for
20: Filter out the spatial objects encrypted with $SEK_P$ in $\mathbb{R}'$
21: for each spatial object $S \in \mathbb{R}'$ do
22:     Verify $S$ with its signature
23:     if $S$ has an invalid signature then
24:         Report the anomaly to the client and exit
25:     end if
26: end for
27: if $\mathbb{C} \neq \mathbb{R}'$ then
28:     Report the anomaly to the client and exit
29: end if
30: Return $\mathbb{R}$
```

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 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.

Figure 4 illustrates an example of k-nearest-neighbor query integrity auditing. The client first encodes the location of the query point with SEK_P and computes that 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 Fig. 4a. 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 a Hilbert value 32; we can then acquire 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, the exact query result being shown in Fig. 4b. The QIAKNN algorithm is formalized in Algorithm 2.



(a) Finding the search upper bound. (b) Searching k nearest objects of Q.

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

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

```
1:  Compute $V_{\mathcal{H}} = \mathcal{T}(x_Q, y_Q)$ based on $SEK_P$
2:  Set $\delta$ = true, $\lambda$ = 1, and $\gamma$ = 0
3:  while $\delta$ do
4:      Set $\mathbb{R} = \emptyset$
5:      $k' = k(1 + \lambda * r)$
6:      $\mathbb{R} \cup$ Retrieve $k'$ objects closest to Hilbert value $V_{\mathcal{H}}$ from the server
7:      Filter out the spatial objects encrypted with $SEK_S$ in $\mathbb{R}$
8:      if $|\mathbb{R}| \geq k$ then
9:          $\delta$ = false
10:     else
11:         $\lambda = \lambda + 1$
12:     end if
13: end while
14: for $i = 0$; $i < |\mathbb{R}|$; $i$++ do
15:     if $Dist(Q, s_i) > \gamma$ then
16:         $\gamma = Dist(Q, s_i)$
            /* $s_i \in \mathbb{R}$ */
17:         $s^* = s_i$
18:     end if
19: end for
20: Compute the edge length of $Q_R$ by $Dist(Q, s^*)$
21: $\mathbb{R}'$ = QIARQ($Q_R$)
22: Set $\gamma = \infty$, $\lambda = 0$, and $\mathbb{R} = \emptyset$
23: for $j = 0$; $j < |\mathbb{R}'|$; $j$++ do
24:     if $\lambda < k$ then
25:         $\lambda = \lambda + 1$
26:         $\mathbb{R} = \mathbb{R} \cup s_j$
27:     else
28:         sort $\mathbb{R}$ 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
```

5.4 Attack-aware auditing query composition

The purpose of our dual space encryption design is to allow for sophisticated cross examination. The client carries out cross examination against a single spatial database that has two different encryptions. However, negligent auditing queries launched by the client may reveal critical information, which could allow malicious LBS providers to detect the correspondence among the data with different encryption keys. For example, if one assumes the 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 results of both queries to jeopardize future queries without being detected by the client.

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 in order to evaluate the integrity of multiple queries. Because queries launched by the same mobile user usually exhibit spatial locality [15, 16], the query range overlap is significant between successive queries from an identical user. By evaluating the integrity of multiple queries from a few users at a time, we can improve security and decrease the integrity auditing overhead of the trusted gateway. The trusted gateway 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, based on the memory capacity and maximum tolerable delay. Only queries whose results contain replicas should be included in $\mathcal{Q}$.

6 Freshness guarantee mechanism

In addition to audit query integrity, another challenging issue of supporting spatial data outsourcing lies in providing freshness guarantees. For outsourced databases, freshness means that all database update operations have been executed correctly by the service provider. In this research, we focus on auditing SQL UPDATE operations in outsourced spatial databases since point-of-interest datasets have more UPDATE operations than INSERT or DELETE operations for supporting location-based applications. For example, the database owner may update the gas price information of each gas station in the outsourced spatial database everyday; however, gas station records are rarely inserted or deleted.

6.1 Freshness auditing for UPDATE operations

The goal of freshness auditing is to assure that queries are executed with the up-to-date database rather than some version of the database in the past. Our approach is based on deterministic functions and time stamp information which are shared by the database owner and the trusted gateway. Specifically, deterministic functions decide when and where (spatial regions) to update the time stamps of records (time stamps store the latest update time of records). For example, assume that a malicious service provider always ignores update requests from the data owner. The client is able to discover the dishonesty because they know the latest update time of certain records by running the deterministic functions.

In order to audit the integrity of UPDATE operations, the database owner submits fake UPDATE statements to the outsourced spatial database. After the outsourced database adopts our mechanism, which is a probabilistic guarantee of freshness, the service provider cannot distinguish fake operations from real operations. A fake UPDATE operation renews the time stamp information (i.e., the T_s field) of all the data records in a given update region of the search space. By checking the T_s field of the data records in the updated region, the client is able to audit the freshness of the outsourced spatial database and tell if the service provider is honest. The generation of fake UPDATE operations is controlled by a group of deterministic functions, which are employed to synchronize the auditing knowledge of the database owner and the trusted gateway [32]. Consequently, a client can join the system anytime and still knows when the next fake UPDATE will happen by executing the deterministic functions.

The major advantage of utilizing deterministic functions to provide freshness guarantees is that the client only needs to store the functions rather than all the spatial regions, which will be updated by the database owner in the future. The functions generate a deterministic sequence of fake UPDATEs at deterministic points of time to renew the time stamps of selected records in the outsourced spatial database. Specifically, the database owner and the trusted gateway share a set of deterministic functions $\mathcal{F}$ and a time slot generation function $\mathcal{P}$ which are detailed as follows:

- $\mathcal{F}$ is a deterministic function set, $\mathcal{F} = \{f_1, \ldots, f_n\}$, where each member function f_i is determined by a randomly chosen key (e.g., the latest update time) and f_i returns the center point and side lengths of an update region U_r (a rectangular area). The distribution of update regions is based on the data distribution of the outsourced spatial database. The number of functions in $\mathcal{F}$ is a system parameter determined by users. The ratio between fake updated statements and real update operations provides the confidence level of detecting malicious service providers which skip update requests from clients.
- $\mathcal{P}$ is a function determined by a randomly chosen key, and it generates a deterministic series of time slots $\{t_1, \ldots, t_m, \ldots\}$ for deciding the next fake operation launch time. In particular, let t_i be the last time which the database owner launches fake UPDATE operations. Then, the database owner will submit fake UPDATE statements again at $\mathcal{P}(t_i) = t_{i+1}$. Also, the distribution of fake UPDATEs generated by $\mathcal{P}$ should be similar to the distribution of real UPDATEs from the database owner.

When the system starts, the database owner D_o executes $\mathcal{P}$ to figure out the next time slot t_i in order to submit fake UPDATE statements. At t_i, the database owner runs the functions in $\mathcal{F}$ for generating a group of update regions. For each update region U_r, D_o renews the time stamp field of all the covered data records with t_i. The size of U_r is determined by the data density, and each U_r proportionally covers a certain number of data records. In case an updated region covers no data record, the database owner inserts a fake record (which can be recognized by the client by setting the I_d field as f) with the current t_i value as its time stamp. For fake record deletion, D_o employs two functions, $h(u)$ and $f(u)$, where u is the total amount of time which a

Algorithm 3 Freshness Auditing for UPDATE operations

```
1: $t_i = \mathcal{P}(0)$
2: while true do
3:    if current_slot = $t_i$ then
4:       for each $f_i \in \mathcal{F}$ do
5:          Generate an update region $U_r$
6:          Renew the $T_s$ field of all the data records covered by $U_r$ in the outsourced spatial database
7:          if $U_r$ covers no data record then
8:             Insert a fake data record $S_f$
9:             $S_f.T_S = t_i$
10:         end if
11:      end for
12:      Delete existing fake records based on functions $h(u)$ and $f(u)$
13:      $t_i = \mathcal{P}(t_i)$
14:   else
15:      wait()
16:   end if
17: end while
```

fake record stays in the outsourced database. $h(u)$ is a hash function which maps to [0, 1] in uniform distribution; $f(u)$ is a decreasing function (e.g., $f(u) = 1/(u+1)$) which starts from 1 and eventually converges to 0 when u goes to infinite. D_o memorizes the u value of each existing fake record and employs the product of the two functions to determine the probability P_d of removing a certain fake record at each time slot (i.e., $P_d = 1 - h(u) f(u)$). In addition, D_o can choose any pair of functions with the aforementioned characteristics to serve the purpose. This design makes it difficult for malicious service providers to discover fake records in outsourced databases.

On the other hand, the trusted gateway also possesses $\mathcal{F}$ and $\mathcal{P}$ for auditing the freshness of outsourced spatial databases. During each fake operation time slot, the client executes the functions in $\mathcal{F}$ for obtaining all the update regions valid for the current time slot. Afterward, the client submits freshness auditing queries for retrieving the data records covered by selected update regions and checks their time stamps. If any T_s of the retrieved data records is out of date, the mismatch demonstrates that the service provider is malicious. In practice, auditing queries for freshness guarantees can be combined with auditing queries for range or *k*NN queries for saving resources. The steps of deterministically submitting fake UPDATE operations by D_o are formalized in Algorithm 3.

6.2 Freshness auditing for INSERT and DELETE operations

Occasionally, the database owner inserts or deletes data records in outsourced POI databases (e.g., a newly opened gas station). Consequently, we also design mechanisms for the database owner to audit if the service provider honestly executes INSERT and DELETE operations. During each fake operation time slot t_i, D_o launches a series of auditing range queries Q_A to verify whether the service provider has indeed executed all the INSERT and DELETE statements submitted through t_{i-1}. The query window size of Q_A is randomly decided within a user defined range (e.g., 0.1% of the search space), and the query window covers one or more executed INSERT/DELETE operations based on their spatial distributions.

7 Query integrity assurance analysis

We consider that a malicious server might tamper with the outsourced database and return incorrect query results to a client. The server can modify/delete existing data recording or inject new data records that do not belong to the data owner. However, since the data owner signs every data object in the outsourced database, injected or modified data records can be easily detected by a client via checking the cryptographic signature. This is because forging or altering a signature is computational intractable for a polynomial-time adversary. Consequently, query integrity is guaranteed in the cases of data injection and modification. However, the signature scheme does not protect the integrity of the query result from the following two attack models: (1) data deletion attacks: for example, a malicious server can delete some data objects from the query result and return an incomplete set of the result to the client. Such an attack model usually takes place when the client is not aware of the total number of objects in the result set, such as a range query; (2) data substitution attacks: for example, in a *k*NN query, a client requests for exact *k* objects from a service provider (SP). A malicious SP can substitute the *k*NN result with objects that are more favorable to the server. Since the replacement objects are also part of the original database, they are authentic data objects provable by their signatures. Our dual-encryption scheme is designed to provide protection to both aforementioned attacks by using the dual spatial encryption technique.

We also investigate the query integrity assurance properties of our system. The notations used in this section are summarized in Table 2. Similar to other research works [8, 13] that employ the Hilbert curve as the space encryption technique, we assume that Hilbert curves are secure, and the inversion of the Hilbert values to the original coordinates is computational intractable. Interested readers can refer to [13] for a security proof of the space encryption method using Hilbert curves. A malicious server may perform deletion attacks on queries that do not have a known number of results to the client. In Section 5, we propose two query auditing paradigms that clients can choose between accordingly. In particular, for the one-to-one query and auditing paradigm (*1T1QA*), a client issues an auditing query after each regular query; for the one-to-many query and auditing paradigm (*1TMQA*), a client issues one auditing query for multiple regular queries that it intends to audit. *1TMQA* requires more storage space on the client-side, i.e., caching all the query results to be verified later. However, *1TMQA* is more attack-proof against a malicious server because it is harder to find the correspondence between data objects in the regular query and the auditing query. Note that the algorithms are probabilistic, therefore,

Table 2 Symbolic notations

Symbol	Meaning
$\mathbb{D}$	Outsourced spatial database
Q_R	Spatial queries
Q_A	Auditing queries
r	Database replication percentage
k	Number of objects in a query result set
d	Number of objects deleted/substituted by a malicious server
m	Number of queries covered by an auditing query in the *1TMQA* model which is set by the trusted gateway

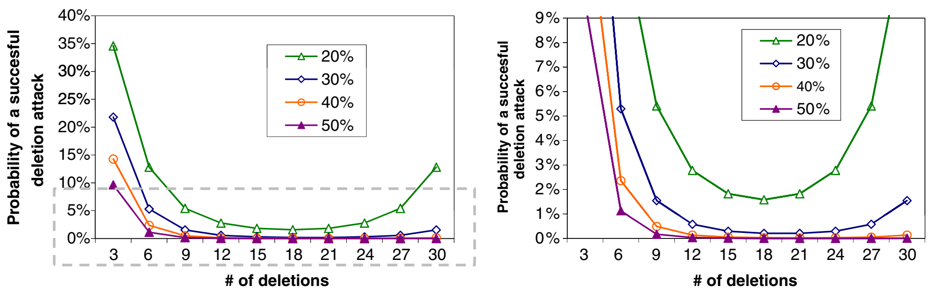
false positives can be produced (i.e., some attacks can escape from being detected). But, it never produces false negatives (i.e., an honest query result will never be considered as dishonest).

We first consider the case of deletion attacks. Before a client sends a range query Q_R to the server, it first converts Q_R into a sequence of Hilbert segments covered by the query range. Then, the set of Hilbert segments are sent to the server. For simplicity, we assume that data points are uniformly distributed in the normalized domain space $[0, 1]^2$. Let the number of objects that fall inside the query range be k. The Hilbert values of the k objects are covered by the Hilbert segments requested by the client. Note that the duplicates of the database may have the same Hilbert values, even though they are encrypted by a different Hilbert curve (SEK_S). Objects that have the same Hilbert values but are not inside the query range (*false hits*) are also returned in the query result. The expected number of these *false hits* is $r \cdot k$. Similarly, for the auditing query, the result contains both the objects in the query range and the objects with the same Hilbert values but not in the query range. Clients can eliminate these objects by their dual information, but the server cannot. The client receives $(k + k \cdot r)$ objects for Q_R and its auditing query Q_A, respectively. In the result of Q_R, $k \cdot r$ objects ($\mathbb{C}$) have their counterparts in the auditing query result, $k \cdot (1 - r)$ objects ($\mathbb{S}$) have not, and $k \cdot r$ objects are false hits to be eliminated by the client. For the auditing query Q_A, $k \cdot r$ objects ($\mathbb{R}'$) are the counterparts of $\mathbb{C}$, and the rest are false hits to the query. For the server to successfully launch a deletion attack, the objects deleted from the two result sets must agree. In other words, if an object is deleted from $\mathbb{C}$, it also has to be deleted from $\mathbb{R}'$. As the server does not know the correspondence between objects, a random selection model is employed here. Therefore, the probability of a successful deletion attack is:

$$Pr(successful\ deletion\ attack) = \frac{\sum_{i=0}^{d}\sum_{j=0}^{d} C_i^{k\cdot(1-r)} \cdot C_j^{k\cdot r} \cdot C_{d-i-j}^{k\cdot r} \cdot C_{d-j}^{k\cdot r}}{(C_d^{k\cdot(1+r)})^2} \qquad (1)$$

As indicated by the equation, the probability of a successful deletion attack is a function of three variables: the selectivity (k) of a query, the number of deletions (d) and the replication ratio (r) of the database. The probability function $Pr(k, d, r)$ is plotted in Figs. 5, 6 and 7.

The relationship between the probability of a successful attack v.s. the number of deletions per query made by a malicious server is shown in Fig. 5. The selectivity of



(a) Probability range from 0% to 40%. (b) Probability range from 0% to 9%.

Fig. 5 The probability of one successful attack v.s. the number of deletions ($k = 30$)

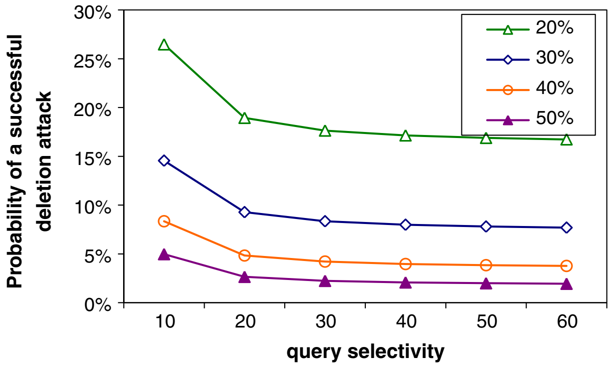


Fig. 6 The probability of one successful attack v.s. the size of result sets ($d = 5$)

the query (k) is 30. To better visualize the small probability values, the part of Fig. 5a in the grey box is magnified in Fig. 5b. The figure demonstrates that as the number of deletions increases, the probability of launching successful attacks quickly decreases. When r is more than 30%, the probability of a successful attack is less than 15% when the server deletes three or four objects. When the number of deletions is close to the total number of objects in the result set, the probability of successful attacks increases (shown as the rising tail in Fig. 5). However, this trend disappears as the replication ratio increases. Notice that if a malicious server returns an empty result set, the client cannot verify whether or not the result is authentic. To solve this problem, in practice, the client only needs to rewrite the auditing query to a kNN ($k = 1$) query in order to verify the distance from the first nearest neighbor to the query point.

Figure 6 shows the relationship between the probability of a successful attack and the selectivity of queries. The number of deletions d is set to 5. As can be seen from the figure, higher replication percentages lead to a lower probability of successful attacks by a malicious server. In general, with 30% replication of the original database, a client can correctly detect more than 85% of the attacks.

Figure 7 illustrates the relationship between the probability of a successful attack and the number of deletions per query. The replication ratio (r) is 30%. The figure shows again that the more objects a malicious server deletes, the higher the

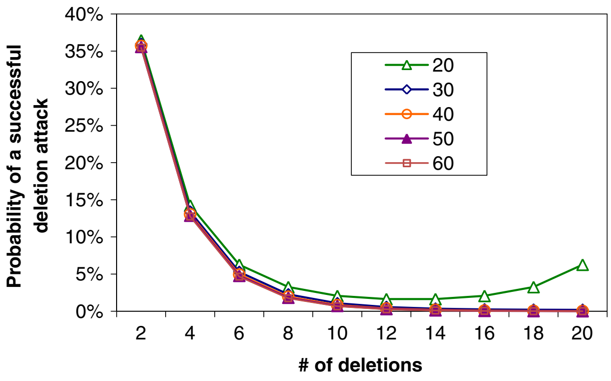


Fig. 7 The probability of one successful attack v.s. the number of deletions ($r = 30\%$)

probability of a client detecting corrupted results. This trend remains as the query selectivity increases.

When a client employs the *1TMQA* model, the probability of a successful attack is:

$$Pr(successful\ deletion\ attack) = \frac{\sum_{i=0}^{d}\sum_{j=0}^{d} C_i^{k\cdot(1-r)} \cdot C_j^{k\cdot r} \cdot C_{d-i-j}^{k\cdot r} \cdot C_{d-j}^{m\cdot k\cdot r}}{(C_d^{k\cdot(1+r)}) \cdot (C_d^{m\cdot k\cdot(1+r)})} \quad (2)$$

The variables k, d and r are the same as in Eq. 1. m is the number of queries covered by one auditing query in the *1TMQA* model. For substitution attacks, a malicious server replaces objects with objects that does not belong to the query result. For example, when a client asks for the k nearest restaurants to its current location, the malicious server deletes restaurants that are close by and returns restaurants that are further away. The probability of a successful substitution attack is: $Pr(successful\ substitution\ attack) = (1-r)^d \cdot Pr(successful\ deletion\ attack)$, where $(1-r)^d$ is the probability of all the substituting objects not having counterparts.

8 Experimental validation

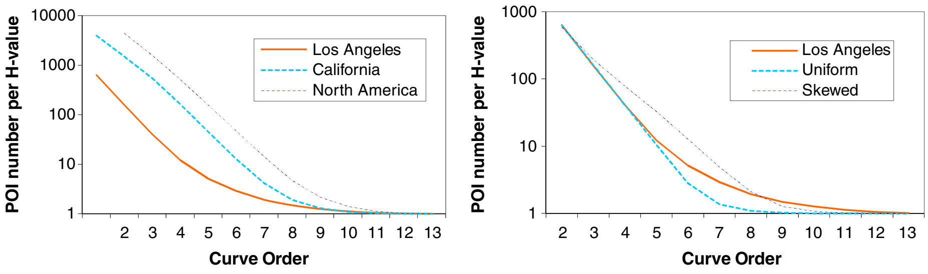
In this section, Hilbert curves are used as the space encoding technique to encrypt spatial information in outsourced databases. It has been proven in [13] that without knowing the space encryption key, a brute force attack will need to exhaustively search all possible key combinations. The complexity 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 encryption function in our design. Table 3 illustrates two synthetic datasets and three real-world datasets utilized in these 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 and North America, respectively. We implemented our query integrity assurance algorithms in Java and conducted the experiments 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.

8.1 Encoded POI density

We first show the relationship between Hilbert curve orders and the number of POIs encoded by one Hilbert value (POI density). Using a higher Hilbert curve order increases the security level of the corresponding space encryption key, while a higher

Table 3 The simulation datasets

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



(a) POI density on different size of datasets. (b) POI density on different data distributions.

Fig. 8 The relationship between curve order and POI density

curve order incurs higher computational complexity. As shown in Fig. 8, the number of POIs per Hilbert value decreases rapidly as the curve order increases on both synthetic and real-world datasets. The average number of POIs per Hilbert value is close to 1 when the curve order reaches 12. Hence, we use the default curve order of 12 for space encryption unless explicitly specified.

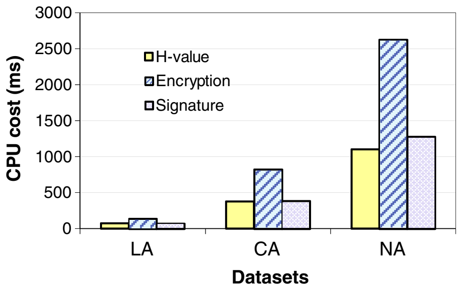
8.2 Spatial database outsourcing initialization

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 operations (2) and (3), and the cost of each may vary. We used the Blowfish encryption algorithm [27] and MD5 (Message-Digest algorithm 5) for signature computation. We perform the data initialization process on all the three real-world datasets with a 40% duplication rate and a curve order of 12. The cost of each operation in Fig. 9 shows that computing the Hilbert values is efficient and less expensive than the other two operations.

8.3 Query translation on the client side

For range queries, a client translates the range query window to the Hilbert curve segments and sends these segments to the service provider. A *k*NN query, as described in Algorithm 2, is split into two parts: retrieving *k* nearest POIs simply

Fig. 9 Initialization cost of the proposed spatial database outsourcing approach



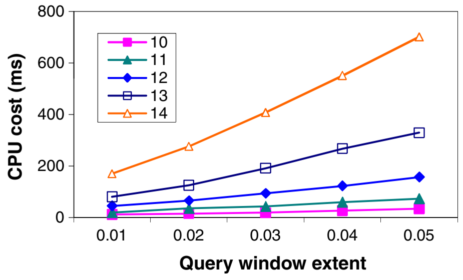


Fig. 10 Query translation cost on the client side

based on the Hilbert value of the query point and launching a range query using the distance of the k^{th} point in the previous operation as the search upper bound.

The query translation cost on the client side is studied in this set of experiments. The size of the range query window varies from 0.01 to 0.05 on the normalized dataset and the curve order varies from 10 to 15. Figure 10 demonstrates the average cost of 50 range queries on the Los Angeles dataset. The figure shows that query translation cost increases when we increase the order of the Hilbert curves. Furthermore, as the range query window increases, the query translation cost also increases because more Hilbert segments are covered by the query window. The query translation cost is measured by computing the Hilbert segments for each query from scratch. In practice, however, the client can maintain a local cache of the mapping from locations to Hilbert values in order to reduce the query translation cost.

8.4 Integrity auditing

Clients employ QIARQ and QIAKNN to verify that 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 on a chosen encryption method. Therefore, we only show the cost of (3) in Fig. 11 for range queries at different range extents and data replication percentages. Verifying query results is efficient when the size of objects returned is small. For location-based

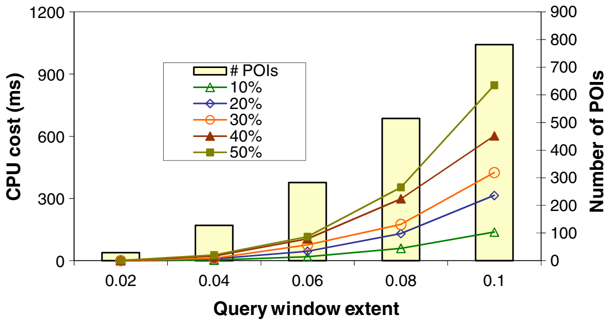
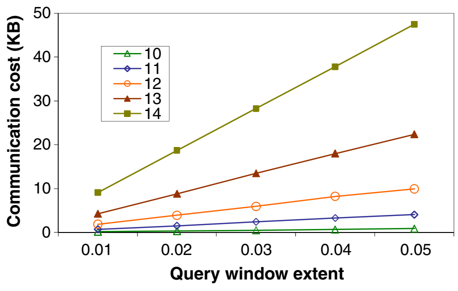


Fig. 11 Cost of query verification

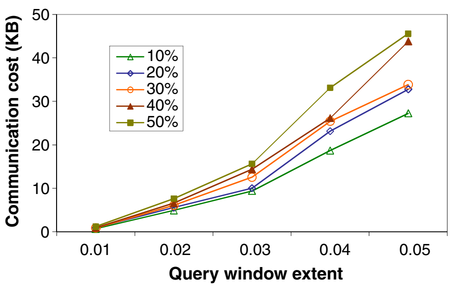
spatial queries, clients are normally only interested in a relatively small region and retrieve a few points of interest in the nearby neighborhoods. Therefore, most of the spatial queries, in practice, are small queries. Figure 11 shows the query verification cost v.s. the size of the query window. To give a better understanding of how large the query windows are, the number of objects enclosed in the query range is also shown in Fig. 11 as columns. The experimental results demonstrate a slow start with a low verification cost while the query size remains small. As the query range increases, query verification becomes costly. However, we argue that in real-world applications, a query size of 100 objects is generally enough for most location-based services. Therefore, our technique is practical for such queries.

8.5 Communication cost

In the next set of experiments, we study the communication cost measured by the size of the data transferred between the client and the service provider per query. 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. The query transfer from a client to a server is composed of several Hilbert curve segments, the size of which is determined by the Hilbert curve order of the SEK. The result returned back from the service provider contains all the points of interest with respect to the query; the size is determined by the distribution of POIs in the database, 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 Fig. 12a, the communication cost is measured by the number of segments transmitted from a query client to the server. In Fig. 12b, the cost is measured by the number of objects returned to the query client. The experiments were performed on the Los Angeles dataset and the trend was similar in the other datasets. It is shown that the communication cost increases linearly as the query window increases on both figures. Furthermore, the results show that the client-to-server communication cost rises as the order of the Hilbert curve chosen increases. However, the server-to-client communication cost is not related to the curve order, as it is dominated by the size of the query.



(a) Client-to-server communication cost.



(b) Server-to-client communication cost.

Fig. 12 The cost of communication between a client and a service provider

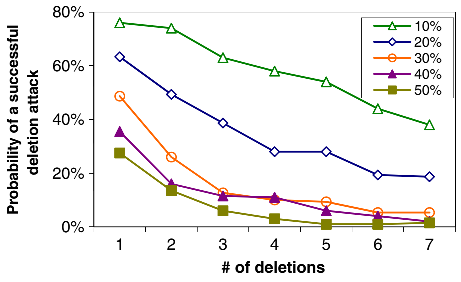


Fig. 13 Probability of escaping detection of deletion attacks

8.6 Query integrity protection against deletion attacks

We discussed in Section 7 that modifying and adding data objects to an outsourced spatial database can be easily detected by our integrity auditing algorithms, which result in one of the two cases: an inability to perform decryption on the tampered data, or inconsistent cryptographic signatures. Consequently, the attack model studied in this set of experiments focuses on data objects deleted by malicious service providers. We conducted the experiments on the Los Angeles dataset using randomly generated queries with the extent of 0.04 on the normalized coordinates. The malicious server launches deletion attacks on query results randomly; the number of deletions varies from 1 to 7. Figure 13 shows the probability that the attacker can successfully escape from being detected by the client versus the number of data objects deleted from a query. It is shown in this figure that, with a replication rate of 30%, the probability of a successful attack is less than 15% when the server deletes three or more objects. Our experimental results agree with the formal analysis in Section 7.

8.7 Query freshness auditing

As discussed in Section 6, freshness guarantee is also an important factor in providing query integrity to all query clients. We proposed a query freshness auditing algorithm to empower clients to verify that query results from the server are up-to-date. In the last set of experiments, we evaluate the effectiveness of the algorithm with regard to the probability of an update attack escaping from being detected. As the update

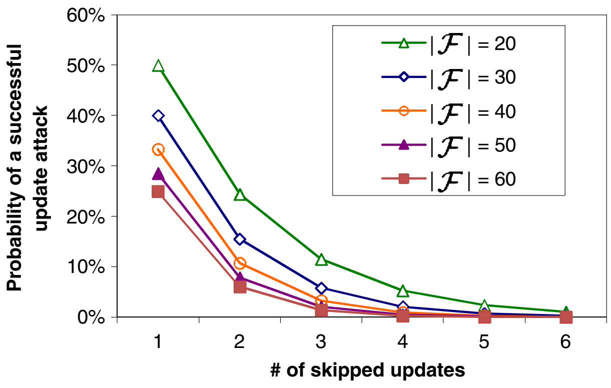


Fig. 14 Probability of escaping detection of skipping updates

information are all encrypted, the service provider cannot tell real updates from fake updates. Consequently, we assume a random attack model from the perspective of the service provider. That is, the service provider drops update requests from the data owner (DO) at random. In Fig. 14, we show the probability of successful update attacks. The x-axis represents the number of updates skipped by the service provider and y-axis shows the probability of successful update omissions without being detected by a client using our algorithm. $\mathcal{F}$ is a deterministic function set and $|\mathcal{F}|$ represents the number of functions contained in the set. In this set of experiments, the update frequency from the data owner is set to 20 records per time slot. It is shown that our algorithm can detect any update omission with a probability greater than 75% when the number of functions in $\mathcal{F}$ is greater than 60. The probability increases quickly as the number of skipped updates increases. For example, when there are two update omissions, the probability of detecting such an attack is increased to 95% with $|\mathcal{F}|$ set to 60. Moreover, as shown in Fig. 14, the more functions $\mathcal{F}$ contains, the less likely that an update omission will escape from being detected by a client.

9 Conclusion

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 either for data privacy protection or for query integrity auditing instead of considering both data privacy and query integrity in tandem. We have introduced query integrity assured algorithms for both range queries and *k*-nearest-neighbor queries with space encryption techniques to secure data privacy. In addition, we have presented mechanisms to provide freshness guarantees for outsourced spatial databases. We have demonstrated through theoretical analyses and simulation results that our mechanisms perform remarkably. For future work, we plan to extend our algorithms to support more spatial query types such as spatial join, spatial path queries, etc.

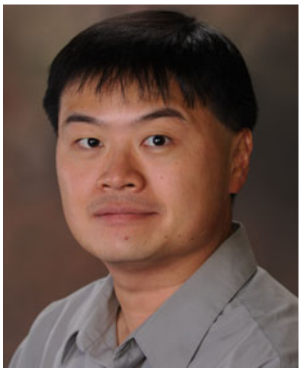
Acknowledgements This research has been funded in part by the US National Science Foundation (NSF) grants IIS-0238560 (PECASE), IIS-0534761, IIS-0742811, CNS-0831502 (CT), and CNS-0831505 (CT), CNS-0855251 (CRI), UT-Battelle grant (UT-B-4000087151), and in part from the METRANS Transportation Center, 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. Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order-preserving encryption for numeric data. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 563–574
2. Butz AR (1971) Alternative algorithm for Hilbert's space-filling curve. IEEE Trans Comput 20(4):424–426
3. Cachin C, Schunter M (2011) A cloud you can trust. IEEE Spectrum 48(12):28–51
4. Devanbu PT, Gertz M, Martel CU, Stubblebine SG (2000) Authentic third-party data publication. In: Proceedings of the 14th annual working conference on Database Security (DBSec), pp 101–112
5. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inf Theory 22(6):644–654

6. Dolan B (2011) 10 reasons why Google Health failed. Mobihealth News
7. Ferdowsi A (2011) Yesterday's Authentication Bug. http://blog.dropbox.com/?p=821. Accessed 5 Mar 2012
8. Ghinita G, Kalnis P, Skiadopoulos S (2007) Prive: anonymous location-based queries in distributed mobile systems. In: WWW, pp 371–380
9. Hacigümüs H, Iyer BR, Li C, Mehrotra S (2002) Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 216–227
10. Hacigümüs H, Mehrotra S, Iyer BR (2002) Providing database as a service. In: Proceedings of the 18th International Conference on Data Engineering (ICDE), p 29
11. Hilbert D (1891) Ueber die stetige Abbildung einer Linie auf ein Flchenstck. Math Ann 38:459–460
12. Jagadish HV (1990) Linear clustering of objects with multiple atributes. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 332–342
13. Khoshgozaran A, Shahabi C (2007) Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In: Proceedings of the 10th international Symposium on Spatial and Temporal Databases (SSTD), pp 239–257
14. Ku W-S, Hu L, Shahabi C, Wang H (2009) Query integrity assurance of location-based services accessing outsourced spatial databases. In: Proceedings of the 11th international Symposium on Spatial and Temporal Databases (SSTD), pp 80–97
15. Ku W-S, Zimmermann R, Wang H (2007) Location-based spatial queries with data sharing in wireless broadcast environments. In: ICDE, pp 1355–1359
16. Ku W-S, Zimmermann R, Wang H (2008) Location-based spatial query processing in wireless broadcast environments. IEEE Trans Mob Comput 7(6):778–791
17. Lawder JK, King PJH (2001) Querying multi-dimensional data indexed using the Hilbert space-filling curve. SIGMOD Record 30(1):19–24
18. Merkle RC (1989) A certified digital signature. In: Proceedings of the 9th annual international cryptology conference (CRYPTO), pp 218–238
19. Moon B, Jagadish HV, Faloutsos C, Saltz JH (2001) Analysis of the clustering properties of the Hilbert space-filling curve. IEEE Trans Knowl Data Eng 13(1):124–141
20. Mouratidis K, Sacharidis D, Pang H (2009) Partially materialized digest scheme: an efficient verification method for outsourced databases. VLDB J 18(1):363–381
21. Mykletun E, Narasimha M, Tsudik G (2004) Authentication and integrity in outsourced databases. In: Proceedings of the Network and Distributed System Security Symposium (NDSS)
22. Orenstein JA (1986) Spatial query processing in an object-oriented database system. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 326–336
23. Pang H, Jain A, Ramamritham K, Tan K-L (2005) Verifying completeness of relational query results in data publishing. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 407–418
24. Pang H, Zhang J, Mouratidis K (2009) Scalable verification for outsourced dynamic databases. Proceedings of the VLDB Endowment (PVLDB) 2(1):802–813
25. Papadopoulos S, Papadias D, Cheng W, Tan K-L (2009) Separating authentication from query execution in outsourced databases. In: Proceedings of the 25th International Conference on Data Engineering (ICDE)
26. PlayStation Network hack: why it took Sony seven days to tell the world (2011) http://www.guardian.co.uk/technology/gamesblog/2011/apr/27/playstation-network-hack-sony. Accessed 5 Mar 2012
27. Schneier B (1994) Description of a new variable-length key, 64-bit block cipher (Blowfish). In: Fast Software Encryption, Cambridge Security Workshop, pp 191–204. Springer, London, UK
28. Schneier B (1996) Applied cryptography (2nd ed). Protocols, algorithms, and source code in C. Wiley, New York, NY, USA
29. Sion R (2005) Query execution assurance for outsourced databases. In: Proceedings of the 31st international conference on Very Large Data Bases (VLDB), pp 601–612
30. Sommerville I (2006) Software engineering (8th edn). Addison Wesley
31. Wang H, Yin J, Perng C-S, Yu PS (2008) Dual encryption for query integrity assurance. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM), pp 863–872

32. Xie M, Wang H, Yin J, Meng X (2008) Providing freshness guarantees for outsourced databases. In: Proceedings of the 11th international conference on Extending Database Technology (EDBT), pp 323–332
33. Yang Y, Papadopoulos S, Papadias D, Kollios G (2008) Spatial outsourcing for location-based services. In: Proceedings of the 24th International Conference on Data Engineering (ICDE), pp 1082–1091
34. Yiu ML, Ghinita G, Jensen CS, Kalnis P (2009) Outsourcing of private spatial data for search services. In: Proceedings of the 25th International Conference on Data Engineering (ICDE)
35. Yiu ML, Ghinita G, Jensen CS, Kalnis P (2010) Enabling search services on outsourced private spatial data. VLDB J 19(3):363–384



Wei-Shinn Ku received his Ph.D. degree in computer science from the University of Southern California (USC) in 2007. He also obtained both the M.S. degree in computer science and the M.S. degree in Electrical Engineering from USC in 2003 and 2006, respectively. He is an Associate Professor with the Department of Computer Science and Software Engineering at Auburn University. His research interests include spatial and temporal data management, mobile data management, geographic information systems, and location-based services. He has published more than 60 research papers in refereed international journals and conference proceedings. He is a member of the ACM and the IEEE.

Ling Hu is a Ph.D. candidate in the computer science department at the University of Southern California. Her research interests include spatial and temporal data management, geographical information systems, query integrity and query optimization, and data warehousing. She owns an M.S. degree in computer science from Northeastern University. She is a student member of the ACM and the IEEE. Contact her at lingh@usc.edu.

Cyrus Shahabi is a Professor and the Director of the Information Laboratory (InfoLAB) at the Computer Science Department and also the Director of the NSF's Integrated Media Systems Center (IMSC) at the University of Southern California. He is also the CTO and co-founder of a USC spin-off, Geosemble Technologies. He received his B.S. in Computer Engineering from Sharif University of Technology in 1989 and then his M.S. and Ph.D. Degrees in Computer Science from the University of Southern California in May 1993 and August 1996, respectively. He authored two books and more than hundred-fifty research papers in the areas of databases, GIS and multimedia. He was an Associate Editor of IEEE Transactions on Parallel and Distributed Systems (TPDS) from 2004 to 2009. He is currently on the editorial board of the VLDB Journal and IEEE Transactions on Knowledge and Data Engineering (TKDE). He is the founding chair of IEEE NetDB workshop and also the general co-chair of ACM GIS 2007, 2008 and 2009. He chaired the nomination committee of ACM SIGSPATIAL for the 2011–2014 terms. He regularly serves on the program committee of major conferences such as VLDB, ACM SIGMOD, IEEE ICDE, ACM SIGKDD, and ACM Multimedia. Dr. Shahabi is a recipient of the ACM Distinguished Scientist award in 2009, the 2003 U.S. Presidential Early Career Awards for Scientists and Engineers (PECASE), and the NSF CAREER award in 2002.

Haixun Wang is currently a lead researcher at Microsoft Research Asia. He received the B.S. and the M.S. degree, both in computer science, from Shanghai Jiao Tong University in 1994 and 1996. He received the Ph.D. degree in computer science from the University of California, Los Angeles in 2000. He has published more than 100 research papers in referred international journals and conference proceedings. He is a member of the ACM, the ACM SIGMOD, the ACM SIGKDD, and the IEEE Computer Society. He has served in program committees of international conferences and workshops, and has been a reviewer for some leading academic journals in the database field.