Verifying Spatial Queries using Voronoi Neighbors

Ling Hu
Computer Science
Univ. of Southern California
Los Angeles, CA, USA
lingh@usc.edu

Wei-Shinn Ku
Dept. of Computer Science
and Software Engineering
Auburn University
Auburn, AL, USA
weishinn@auburn.edu

Spiridon Bakiras
Dept. of Mathematics and
Computer Science
John Jay College, CUNY
New York, NY, USA
sbakiras@jjay.cuny.edu

Cyrus Shahabi
Computer Science
Univ. of Southern California
Los Angeles, CA, USA
shahabi@usc.edu

ABSTRACT

With the popularity of location-based services and the abundant usage of smart phones and GPS enabled devices, the necessity of outsourcing spatial data has grown rapidly over the past few years. Nevertheless, in the database outsourcing paradigm, the authentication of the query results at the client remains a challenging problem. In this paper, we focus on the Outsourced Spatial Database (OSDB) model and propose an efficient scheme, called *VN-Auth*, that allows a client to verify the correctness and completeness of the result set. Our approach can handle both k nearest neighbor (kNN) and range queries, and is based on neighborhood information derived by the Voronoi diagram of the underlying spatial dataset. Specifically, upon receiving a query result, the client can verify its integrity by examining the signatures and exploring the neighborhood of every object in the result set. Compared to the current state-of-the-art approaches (i.e., methods based on Merkle hash trees), VN-Auth produces significantly smaller verification objects ($\mathcal{VO}$) and is more computationally efficient, especially for queries with low selectivity.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Application—*spatial databases and GIS*

General Terms

Algorithms, Experimentation

Keywords

Spatial database outsourcing, Query integrity, Location-based services, Authentication

1. INTRODUCTION

The amount of information generated in our daily lives has grown rapidly over the past decade. This large amount of information as well as the complexity of the data, demand sophisticated management systems that are beyond the capabilities of many small businesses or individuals. Additionally, the cost of running a state-of-the-art database management system may be significant, far exceeding the initial data acquisition cost. Consequently, the *database outsourcing* paradigm is becoming increasingly popular, and has received a lot of attention in the research community. In this paradigm, the data owner (DO) delegates the management and maintenance of its database to a third-party service provider (SP), and the SP is responsible for indexing the data and answering client queries.

In this work, we focus on the Outsourced Spatial Database (OSDB) model, as shown in Figure 1. We assume that the clients are mobile users who issue location-based queries (e.g., kNN or range queries), in order to discover points of interest (POIs) in their neighborhood. However, since the SP is not the real owner of the data, *query integrity assurance* is an important (and challenging) problem that has to be addressed. In particular, the SP has to prove to the client that (i) the data is originated from the DO and, (ii) the result set is correct and complete.

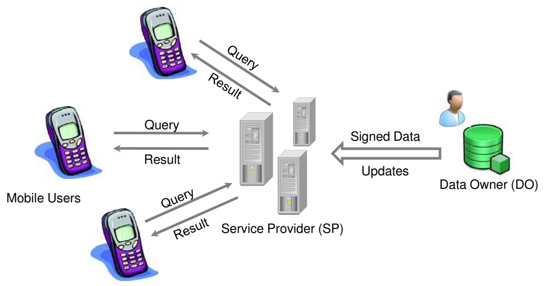


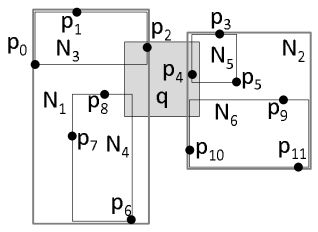
Figure 1: System architecture.

The general framework that is commonly used in the literature is based on digital signatures and utilizes a public-key cryptosystem, such as RSA [5]. Initially, the DO obtains, through a trusted key distribution center, a *private* and a *public* key. The private key is kept secret at the DO, while the public key is accessible by all the clients. Using its private key, the DO digitally signs the data, by generating a number of signatures. Then, it sends the signatures and the data to the SP, which constructs the necessary data structures for efficient query processing. When the SP receives a query from a client, it generates a *verification object* ($\mathcal{VO}$) that contains the result set along with the corresponding authentication information. Finally, the SP sends the $\mathcal{VO}$ to the client, which can verify the results using the public key of the owner.

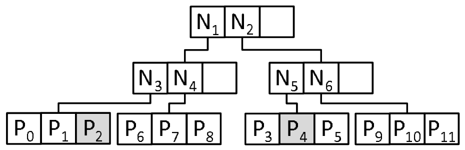
Currently, the state-of-the-art solution for authenticating spatial queries is the Merkle R-tree (MR-tree) [29]. The MR-tree is essentially an R-tree that is augmented with authentication information (i.e., hash digests). In particular, every leaf node of the tree stores a digest that is computed on the concatenation of the binary representation of all objects in the node. Internal nodes are assigned a digest that summarizes the child nodes' MBRs (minimum bounding rectangles) and digests. Digests are computed in a bottom-up fashion, and the single digest at the root is signed by the DO. Range queries on the MR-tree are handled by a depth-first traversal of the tree. The resulting $\mathcal{VO}$ contains (i) all the objects in every leaf node visited, and (ii) the MBRs and digests of all the pruned nodes. Having this information, the client can reconstruct the root digest and compare it against the one that was signed by the owner. In addition, the client also examines the spatial relations between the query and each object/MBR included in the $\mathcal{VO}$, in order to verify the correctness of the result.

We argue that the structure of the MR-tree as well as the verification process, suffer from several drawbacks. First, the authentication information (hash digests) embedded in the MR-tree reduces the node fanout, leading to more I/O accesses during query processing. Second, in the presence of updates, all the digests on the path from an *affected* leaf node to the root have to be recomputed. Consequently, when updates are frequent, query performance is degraded, as discussed in [21]. Finally, the overhead of the $\mathcal{VO}$ can be significant, especially for queries that return only a few objects. This is due to the fact that the SP has to return all the objects that lie inside the leaf nodes that are visited during query processing. As an example, consider the range query q in Figure 2. Even though the result set includes only two objects (p_2, p_4), the corresponding $\mathcal{VO}$ has to return all 12 objects in the database. An extension of the MR-tree, called MR*-tree [30], mitigates this last drawback, by ordering the entries of each node and constructing hierarchical relationships of the digests therein. Nevertheless, it does not eliminate the $\mathcal{VO}$ overhead entirely, while at the same time it increases the verification cost at the client.

Motivated by the above observations, we propose VN-Auth, a novel approach that authenticates arbitrary spatial queries based on neighborhood information derived by the Voronoi diagram of the underlying spatial dataset. In particular, before delegating its database to the SP, the owner transforms each data object by creating a signature of the object itself along with information about its Voronoi neighbors. A key aspect of our method is that it separates the authentication information from the spatial index. As a result, the efficiency of the spatial index is not compromised, and all updates are restricted in the neighborhood of the affected objects. Furthermore, the $\mathcal{VO}$ is extremely compact, since it *only* includes the transformed objects that belong to the result set. Our simulation results show that, compared to the MR-tree variants, VN-Auth produces significantly smaller verification objects and is more computationally efficient, especially for queries with low selectivity.



(a) Points and MBRs



(b) MR-tree

Figure 2: Range query on the MR-tree.

The remainder of the paper is organized as follows. Section 2 reviews the related work, while Section 3 discusses Voronoi diagrams and signature aggregation techniques. Section 4 describes the data transformation process, and Section 5 introduces the verification algorithms for kNN and range queries. Database updates are handled in Section 6, followed by our experimental results in Section 7. Section 8 introduces some additional features of VN-Auth and, finally, Section 9 concludes the paper with directions for future work.

2. RELATED WORK

Section 2.1 reviews query authentication methods with a focus on spatial database outsourcing. Section 2.2 discusses spatial query processing techniques on R-trees.

2.1 Query Authentication for Outsourced Spatial Databases

The idea of outsourcing databases to a third-party service provider was first introduced by Hacigümüs et al. [8]. Since then, numerous query authentication solutions have been proposed for auditing query results in outsourced relational databases [7, 15, 20, 25, 19, 13, 28, 27]. The first mechanism for verifying query results in *multi-dimensional* databases was proposed in [2]. The idea is to add authentication information into a spatial data structure, by constructing certified chains [19] on the data points within each partition as well as on all the partitions in the data space. For a given range query, this approach generates a proof that every data point (within the intervals of the certified chains that overlap the query window) is either returned as a result or falls outside the query range. Based on [2], Cheng and Tan designed a mechanism for authenticating k nearest neighbor (kNN) queries on multi-dimensional databases, ensuring that the result set is complete, authentic, and minimal [3, 4]. Nevertheless, both solutions incur significant authentication overhead, and the required verification information consumes considerable client-server communication bandwidth.

Yang et al. [29, 30] introduced the MR- and MR*-trees, which are space-efficient authenticated data structures supporting fast query processing and verification. The MR-tree augments the standard R-tree, by computing hash digests on the concatenation of the binary representation of all the entries in a tree node. To verify the correctness and completeness of range query results, the generated $\mathcal{VO}$ includes (i) all the visited objects, and (ii) the MBRs and digests of all the pruned nodes. The MR*-tree improves MR-tree by ordering the entries of each node and constructing hierarchical relationships of the digests therein. Entries are sorted according to an in-order traversal of a KD-tree. As a result, when a query intersects an MBR, not all entries are required for query verification, and some of them can be pruned. The idea is similar to building a small Merkle tree on each node of the MR-tree. The MR*-tree reduces significantly the $\mathcal{VO}$ size, but incurs some CPU overhead due to the embedded information. Nevertheless, neither the MR-tree nor the MR*-tree are able to handle data updates efficiently.

Efficient verification in the presence of frequent updates has been studied in the context of relational data. The Partially Materialized Digest scheme (PMD) [14] verifies one-dimensional range queries, and applies to both static and dynamic databases. PMD employs separate indexes for the data and their associated verification information, in order to avoid unnecessary costs when processing queries that do not request verification. Furthermore, Pang et al. [21] introduced a protocol, based on signature aggregation, that verifies the authenticity, completeness and freshness of the query result. An important property of the protocol is that it allows new data to be disseminated immediately, while ensuring that outdated values (beyond a pre-set age) can be detected. In addition, the authors also implemented an efficient verification technique for ad-hoc equi-joins. Papadopoulos et al. [22] designed a solution for the authentication of continuous spatial queries, i.e., queries that are constantly evaluated on a highly dynamic database (consisting of moving objects). The proposed mechanism achieves both correctness and *temporal completeness*, and aims at reducing the transmission overhead between the service provider and the clients.

All the aforementioned solutions require changes to be made in the DBMS kernel, in order to support the embedded authentication information. This may not be realistic in many applications. On the other hand, Ku et al. [12] proposed a query integrity assurance technique that does not require any modifications in the DBMS software. The solution first employs a spatial transformation method that encrypts the spatial data before outsourcing them to the SP. Then, by probabilistically replicating a portion of the data and encrypting it with a different encryption key, clients are able to audit the trustworthiness of the query results. However, since [12] is not a deterministic solution, attacks may escape the auditing process.

2.2 Spatial Query Evaluation

In this paper we focus on two spatial query types, namely k nearest neighbor and range queries, which are the building blocks of most location-based services. With R-tree [6] based spatial hierarchical structures, depth-first search (DFS) [23] and best-first search (BFS) [9] have been the prevalent branch-and-bound techniques for processing nearest neighbor queries. Generally, DFS recursively expands the index nodes for searching nearest neighbor candidates. At each newly visited non-leaf node, DFS computes the ordering metrics for all its child nodes, and utilizes pruning strategies to remove unpromising branches. When a leaf node is reached, the data objects are retrieved and the nearest neighbor candidates are updated.

On the other hand, BFS employs a priority queue Q_p to store all nodes that need to be explored through the search process. Consequently, it supports incremental retrieval of objects when k increases. The nodes in the queue are sorted according to their minimum distance (MINDIST) to the query point. Starting from the root, BFS repeatedly dequeues the top entry in Q_p and enqueues its child nodes with their MINDIST values. When a data entry is dequeued, it is inserted into the result set and the search process terminates when k data objects are retrieved. For range queries that retrieve objects within a user specified region, R-trees provide efficient query processing algorithms as well. Specifically, for a given spatial dataset, the R-tree structure groups objects close to each other into a MBR, and the algorithm only visits the MBRs that overlap with the query range.

3. PRELIMINARIES

Section 3.1 introduces Voronoi diagrams and their properties, while Section 3.2 describes a signature aggregation technique that we utilize in our methods.

3.1 Voronoi Diagrams

Given a set of distinct objects $P = \{p_1, p_2, \ldots, p_n\}$ in $\mathbb{R}^m$, the *Voronoi diagram* of P, denoted as $\mathbb{VD}(P)$, partitions the space of $\mathbb{R}^m$ into n disjoint regions, such that each object p_i in P belongs to only one region and every point in that region is closer to p_i than to any other object of P in the Euclidean space. The region around p_i is called the *Voronoi cell* of p_i, denoted as $VC(p_i)$, and p_i is the *generator* of the Voronoi cell. Therefore, the Voronoi diagram of P is the union of all Voronoi cells $\mathbb{VD}(P) = \{VC(p_1), VC(p_2), \ldots, VC(p_n)\}$. If two generators share a common edge, they are Voronoi neighbors. If we connect all the Voronoi neighbors, we get the *Delaunay triangulation* $\mathbb{DT}(P)$, which is the dual graph of $\mathbb{VD}(P)$. Note that, in this paper, we utilize Euclidean distance functions in $\mathbb{R}^2$.

PROPERTY 1. *Given a set of distinct points $P = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^2$, the Voronoi diagram $\mathbb{VD}(P)$ and the corresponding Delaunay triangulation $\mathbb{DT}(P)$ of P are unique.*

PROPERTY 2. *The average number of Voronoi edges per Voronoi polygon does not exceed six. That is, the average number of Voronoi neighbors per generator does not exceed six.*

PROPERTY 3. *Given the Voronoi diagram of P, the nearest neighbor of a query point q is p, if and only if $q \in VC(p)$.*

PROPERTY 4. *Let $p_1, \ldots, p_k$ be the k ($k > 1$) nearest neighbors in P to a query point q. Then, p_k is a Voronoi neighbor of at least one point $p_i \in \{p_1, \ldots, p_{k-1}\}$.*

The proofs of the above properties are omitted here because of space limit. However, interested readers can refer to [18] (Properties 1-3) and [11] (Property 4) for more details.

3.2 Signature Aggregation

In our approach, the DO generates one signature for every object in the database, which is computed on the hash digest of the concatenation of the binary representation of the object and its Voronoi neighbors. In this way, the client can verify the authenticity of each individual object and its neighborhood. Note that, in this work, we utilize RSA signatures [5] that are typically 128 bytes in size. Alternatively, signatures based on Elliptic Curve Cryptography (ECC) [1] can be significantly shorter, thus reducing the overall communication and storage cost. However, ECC algorithms are computationally intensive, and would perform poorly on mobile devices with limited computational capabilities.

The drawback of having one signature per database object is that it may increase considerably the communication cost between the SP and the client. Specifically, the SP has to transmit one 128-byte signature for every object in the result set, so the overhead can be significant for queries with high selectivity (especially for mobile clients). To avoid this cost, we employ a technique called *signature aggregation*. In particular, given k digests and their corresponding signatures (generated by the same signer), the SP can replace them with a single *Condensed-RSA* signature. Condensed-RSA has the same size as the original signatures (128 bytes), and is computed as the modular multiplication of the k signatures. Aggregate signatures are provably secure [16, 17] and can be computed by any party that possesses the individual signatures.

4. DATA TRANSFORMATION

Consider a DO that has compiled a large collection of n POIs (e.g., restaurants) within a geographic region. Each POI i is represented as a unique object p_i in the database, which has the form $\langle p_i.location, p_i.tail\rangle$. The *location* attribute stores the spatial coordinates of the object, while the *tail* attribute stores some additional information about the object, such as name, address, phone number, web site, etc. Before transmitting the database to the SP, the DO transforms each object by attaching neighborhood and authentication information.

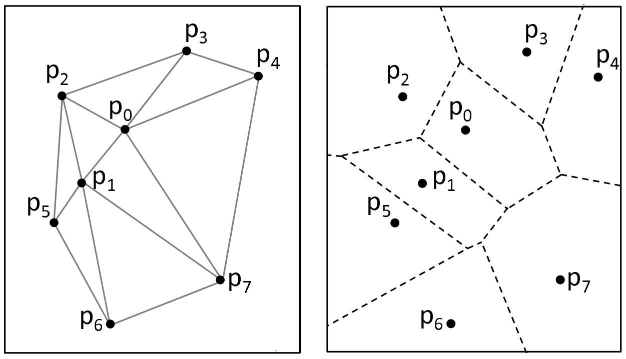
In particular, the DO initially computes the Voronoi diagram of the spatial dataset (as shown in Figure 3) and retrieves the Voronoi neighbors of each POI. Then, it appends a *neighbors* attribute to every object in the database that stores the locations of all its Voronoi neighbors. For example, the *neighbors* attribute of p_5 is equal to:

$$p_5.neighbors = \{3, p_2.location, p_1.location, p_6.location\}$$

Note that, as the number of Voronoi neighbors for a POI is not fixed, the first value of the attribute specifies the exact number of neighbors. Furthermore, we assume that the DO stores the Voronoi neighbors in a clockwise or counter-clockwise order, to facilitate the on-the-fly reconstruction of Voronoi cells at the client (as explained in Section 5.2). The final step is for the DO to sign each individual object, so that the client can verify the authenticity of the information stored therein. Specifically, for an object p_i, its signature $\mathbb{S}$ is computed as

$$\mathbb{S} = sign(h(p_i.location|p_i.tail|p_i.neighbors))$$

where h is a one-way, collision-resistant hash function and '|' denotes the concatenation of two binary strings. To summarize, each transformed object p_i at the DO's site has the form $\langle p_i.location, p_i.tail, p_i.neigbors, p_i.\mathbb{S}\rangle$.



(a) Delaunay triangulation (b) Voronoi diagram

Figure 3: Spatial dataset example.

After the database transformation process completes, the DO transfers all objects to the SP. Upon receiving the database objects, the SP builds an appropriate spatial index, and is then ready for query processing. Note that, the leaf level of the index only stores pointers to the location of the transformed objects on the disk. There are several advantages in our approach, compared to the MR-tree variants. First, the DO is oblivious to the query processing mechanisms at the SP. Consequently, the SP may utilize any spatial index (and query processing algorithm) without informing the DO. Second, the spatial index does not store any authentication information, and thus remains compact and efficient. Third, database updates affect only their local regions, and do not need to propagate to the root of the index. The only drawback of VN-Auth is the storage overhead of the transformed objects. However, according to Property 2 (Section 3.1), the number of neighbors for each generator in the Voronoi diagram does not exceed six, on average. Therefore, the expected overhead per object is $128 + 6 \times 16 + 2 = 226$ bytes (16 bytes/point object and 2 bytes/short integer), which is typically less than the size of the original object.

5. AUTHENTICATING SPATIAL QUERIES

In this section, we introduce the verification algorithms for typical location-based queries. Section 5.1 describes the query processing mechanism at the SP, while Sections 5.2 and 5.3 introduce the verification algorithms for kNN and range queries, respectively.

5.1 Query Processing at the SP

As the authentication information is stored only at the data objects, query processing at the SP is fairly simple. Assuming that a spatial index (e.g., an R-tree) is built on the spatial attributes of the objects, query processing can follow any state-of-the-art algorithm that exists in the literature. Furthermore, when there are more than one objects in the result set, the SP employs the signature aggregation technique to condense the signatures into a single one (as discussed in Section 3.2). Therefore, for each query, the SP returns to the client (i) a list L of objects that belong to the result set, and (ii) an aggregate signature $\mathbb{AS}$.

5.2 kNN Query Verification

Nearest neighbor (NN) queries are the fundamental building blocks in location-based services. In particular, kNN queries allow mobile users to retrieve the k closest POIs from the database, i.e., they may issue queries such as "*find the 10 nearest restaurants to my location*". When a client receives its kNN query result from the SP, it needs to (i) verify the authenticity of the results that all objects are originated from the DO, and (ii) verify the correctness and completeness of the result set. The first task is straightforward, since the client only needs to verify the aggregate signature that is returned by the SP. Note that, forging or altering a signature is computationally intractable for a polynomial-time adversary. For the second task, VN-Auth employs an incremental verification process that is based on Properties 3 and 4. Specifically, according to Property 3, p_i is the 1st NN of the query point q, if and only if q lies inside the Voronoi cell of p_i. Once this geometric test is verified, Property 4 suggests that the 2nd NN of q must be one of the Voronoi neighbors of the 1st NN (p_i). In the general case, the kth NN of a query point q exists in the union of the Voronoi neighbors of the first $(k-1)$ NNs of q.

Algorithm 1 Verify$k$NN(q,L,k, $\mathbb{AS}$)

```
1.  if ($\mathbb{AS}$ is NotValid) then
2.    return false;{signature validation fails}
3.  end if
4.  Min-Heap ← $\emptyset$;
5.  Verified ← $\emptyset$;
6.  VCP ← computeVC($p_1$);
7.  if ($q \notin$ VCP) then
8.    return false;{the 1st NN fails}
9.  else
10.   Verified ← $p_1$;
11. end if
12. for $i = 1$ to $k - 1$ do
13.   for all ($nb \in p_i.neighbors$) do
14.     if (($nb \notin$ Verified) && ($nb \notin$ Min-Heap)) then
15.       Min-Heap ← $nb$;
16.     end if
17.   end for
18.   if ($p_{i+1}.location \neq$ Min-Heap.pop()) then
19.     return Verified; {the $(i+1)^{th}$ NN fails}
20.   else
21.     Verified ← $p_{i+1}$;
22.   end if
23. end for
24. return $L$
```

The kNN query verification process is shown in Algorithm 1. The inputs to the algorithm are the query point q, the result set L, the parameter k, and the aggregate signature $\mathbb{AS}$. We assume that $L = \{p_1, p_2, \ldots, p_k\}$, where p_1 is the 1st NN, p_2 the 2nd NN, and so on. The algorithm first checks the aggregate signature (lines 1–3), and if it is valid an empty min-heap is initiated. Next (lines 6–11), it constructs the Voronoi cell of the first object p_1 and checks if q falls inside $VC(p_1)$. If not, p_1 is not the 1st NN and the verification process fails. Otherwise, p_1 is verified as the 1st NN, and is added to a list of verified objects. Recall that each object is augmented with its Voronoi neighbors in a clockwise (or counter-clockwise) order. Therefore, the Voronoi cell can be computed on-the-fly by finding the circumcenters of the surrounding Delaunay triangles of p_1, which is not an expensive computation and can be performed efficiently on a mobile device.

The subsequent *for* loop (lines 12–23) iterates through all the objects in L and performs the following operations: (i) it inserts the Voronoi neighbors of the last verified object (p_i) into the min-heap, sorted in ascending order of their distances to q (lines 13–17), and (ii) it compares the next object in the result set (p_{i+1}) against the object on the top of the min-heap (lines 18–22). If they are identical, p_{i+1} is verified as the next NN and is added to the list of verified objects. Otherwise, verification fails and the program returns the first i objects in L that were verified successfully. Note that, the capacity of the min-heap is initially set to $(k-1)$, i.e., it will only hold the first $(k-1)$ objects that are closer to q (we are not interested in objects further away). Furthermore, the capacity can be decreased by one after each iteration, in order to minimize the computational and storage cost at the client.

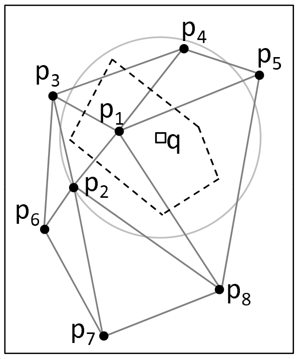


Figure 4: kNN verification example.

To illustrate the kNN verification algorithm, consider the 3NN query q in Figure 4. First suppose that the SP returns the correct result set $L = \{p_1, p_4, p_2\}$. Once the aggregate signature is verified, the client computes the Voronoi cell of p_1 and checks whether q lies inside $VC(p_1)$. Since this is true, p_1 is proven to be the 1st NN of q. Consequently, the algorithm goes through the Voronoi neighbors of p_1 (p_2, p_3, p_4, p_5, p_8) and inserts the two closest objects to q into the min-heap. Therefore, the min-heap is now equal to $\{p_4, p_2\}$. Next, the second object in the result set (p_4) is compared against the object at the top of the heap and, as they are identical, p_4 is verified as the 2nd NN of q. In the next iteration, every neighbor of p_4 is examined, but nothing is inserted into the heap, because p_2 is still the closest point to q (note that the min-heap only contains p_2 now). Finally, p_2 is compared against the 3rd NN reported in L, and L is verified successfully at the client. Suppose now that the SP returns the incorrect result $L = \{p_1, p_4, p_3\}$ to the client, by replacing p_2 with p_3. The aggregate signature is still valid, but the client will discover the error on line 18, as the 3rd NN derived from the first two neighbors should be p_2 rather than p_3.

5.3 Range Query Verification

VN-Auth can also be used to verify arbitrary range queries. Observe that, kNN query verification essentially verifies all the objects inside a circular range centered at the query point q with radius equal to the distance from q to its k^{th} NN. Therefore, this approach can be easily extended to verify circular range queries, such as "*find all restaurants within 200m of my location*". The client first sorts the objects in the result set in ascending order of their distances to the query center, and verifies each object using Algorithm 1. To minimize the computation at the mobile device, the sorting process can be performed at the SP. Note that, after the last object is verified successfully, the client also needs to verify that none of the Voronoi neighbors of the result set lies inside the query range.

Unlike kNN queries that always return exactly k results, circular queries may occasionally return an *empty* set, i.e., no database object may fall inside the query range. We handle such cases by having the SP return the nearest neighbor of query location, i.e., the generator of the Voronoi cell where the query point lies. By exploring the neighborhood of that object, the client can easily verify the absence of objects in the particular range.

Another interesting variation of the range query is a *rectangular* query. Such queries allow clients to specify a rectangular shape on the map and retrieve all POIs that fall inside that area. Square shaped queries are handled trivially, by rewriting the query into a circular one. Figure 5 shows an example of a square query that is replaced by the circumcircle of the square. However, the grey region in Figure 5 may produce some *false hits*, i.e., objects that are returned by the SP but do not belong to the result set (such as object e). We call this region the *overhead region*. The query verification process is identical to the one used for circular queries, with an additional step at the end to filter out the false hits.

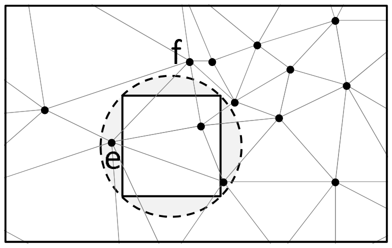
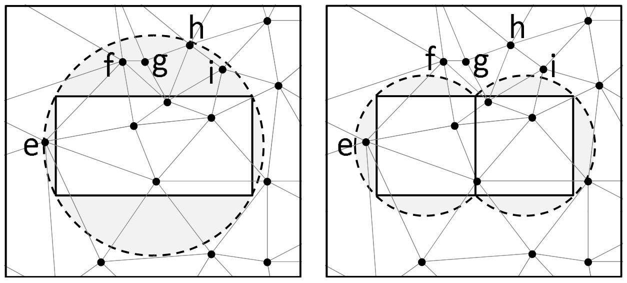


Figure 5: Square query verification.

When rectangular queries are too tall or too wide, the query rewriting mechanism discussed above produces a large overhead region. Therefore, false hits may incur a substantial overhead to the $\mathcal{VO}$, leading to an increased communication and computational cost at the client. Figure 6(a) illustrates this scenario, where five redundant objects (e, f, g, h and i) are included in the result set. To address this drawback, we can rewrite the rectangular query differently. Instead of using one circular range to enclose the whole rectangle, we partition the rectangle into a few squares and generate one circular query per square. In Figure 6(b), for instance, the rectangle is divided into two squares, which eliminates 4 false hits. Note that, the overlapping circular ranges may cause certain objects to be transmitted multiple times to the client. To avoid this cost, the SP may first generate the union of all the result sets, and then send it to the client along with a brief description for each individual result.

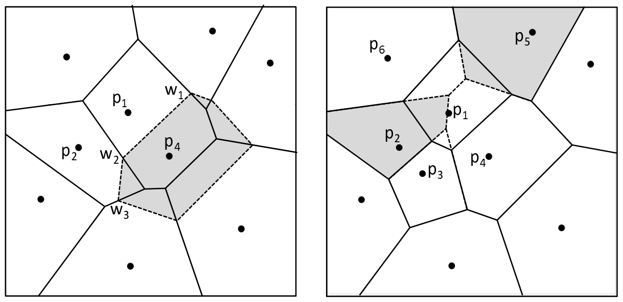


(a) Rectangular query (b) Query rewriting

Figure 6: Minimizing the false hits.

6. DATABASE UPDATES

In this section, we discuss how VN-Auth handles database updates. We only consider object insertions and deletions, since an update is simply the combination of the two operations. As mentioned earlier, one of the drawbacks of the MR-tree based verification techniques is that all updates propagate hash digest recomputations from the leaf nodes to the root. Consequently, frequent updates degrade the performance of the R-tree index and create a performance bottleneck at the root node. Our approach overcomes this problem by separating the authentication information from the spatial indexes. Verification information is attached only to the data objects, and updates affect only a specific neighborhood of the dataset. In addition, as neighboring objects are close to each other, they are often stored on the same or nearby pages on the disk. Therefore, updates in the VN-Auth framework are expected to be more efficient than the MR-tree updates.



(a) Object insertion (b) Object deletion

Figure 7: Object insertion and deletion.

After inserting a new object in the database, the DO employs the *incremental boundary growing* method [18] to compute the updated Voronoi diagram. In Figure 7(a), for instance, to insert a new object p_4, we first identify (i) the Voronoi cell $VC(p_1)$ containing p_4 and (ii) all the Voronoi neighbors of p_1. Then, we draw a bisector between p_1 and p_4, which intersects $VC(p_1)$ at w_1 and w_2. Segment w_1w_2 becomes the first edge of the new Voronoi cell. The next step is to iterate over all the neighboring objects that share Voronoi edges with p_1 in a counter-clockwise fashion (e.g., starting with p_2). Then, we draw the bisector between p_2 and p_4 and retrieve the next intersection at w_3. We repeat this process until all the edges of the new Voronoi cell (of p_4) are computed. The above process can also be performed in a clockwise manner. Finally, the new object is augmented with authentication and neighborhood information, and is transmitted to the SP. In addition to the new object, the owner also transmits to the SP all the objects whose neighborhood information was affected by the inserted object (along with their new signatures). Upon receiving the new object, the SP simply inserts it in the corresponding spatial index.

Deleting an object (e.g., p_1) follows a similar approach, as shown in Figure 7(b). We first locate p_1 and its neighbors, and divide $VC(p_1)$ with the bisectors between the neighboring pairs of p_1. Next, we update the Voronoi neighbors of p_1 with new neighboring information, and transmit all affected objects (with their new signatures) to the SP. Additionally, the SP removes p_1 from the corresponding spatial index.

A final remark concerns *query freshness*, i.e., clients should be able to verify that the database maintained at the SP includes all the updates generated by the DO. A malicious SP may ignore all updates after the database initialization process, but clients will remain oblivious to this fact as the verification process will not be affected. In VN-Auth we follow the methods discussed in [30, 13] that allow signatures to either expire periodically or be selectively revoked by the data owner.

7. EXPERIMENTS

In this section, we evaluate experimentally the performance of VN-Auth, and compare it against the MR-tree variants. Our implementation is in Java, with the client-side application running on a Google Android mobile device, and the server-side (SP) service hosted on a Windows Server PC with Intel Core2 Duo 3GHz CPU and 4GB memory. The DO also runs on a PC with the same configuration. To implement the cryptographic operations, we used the Java cryptography extension packages [10]. Our experiments were performed on two real-world datasets obtained from the U.S. Census Bureau [26]: (i) **CA** which contains 62k data points from California, and (ii) **NA** which consists of 556k POIs taken from North America. Each set of the following experiments was performed on both datasets.

The VN-Auth framework consists of an offline database transformation part, and an online query evaluation part. In the offline phase, we perform a Delaunay triangulation algorithm (Graham scan) to compute the Voronoi neighbors, and then incorporate the verification information into each object. The SP provides online query evaluation to clients after indexing the spatial attributes of the objects with an R*-tree. We also implemented the MR- and MR*-trees, based on the algorithms described in [30], with the R*-tree serving as the basis of both index structures. The page size is set to 4KB for all trees. For both the MR- and MR*-trees, index nodes and leaf nodes have a fan-out of 64 and 256, respectively. For the VoR-tree, index nodes and leaf nodes have a fan-out of 128 and 43 (on average), respectively.

In the first experiment, we investigate the storage overhead required by VN-Auth in order to maintain the neighborhood information of the database objects. Specifically, Figure 8 shows the distribution of the number of Voronoi neighbors per object. For both real-world datasets, over 95% of the objects have less than 10 neighbors and only a few objects have more than 10 neighbors. Clearly, these results confirm Property 2 (see Section 3.1).

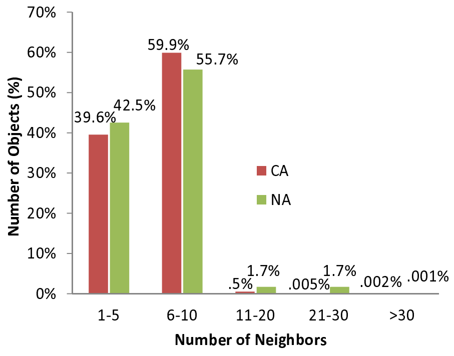
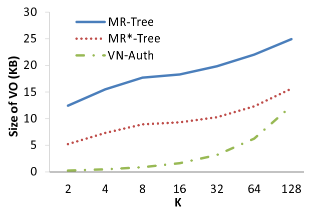


Figure 8: Statistics on the number of neighbors.

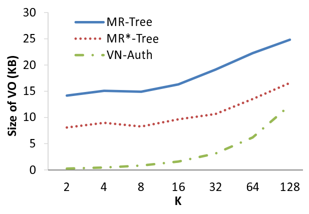
Description	Symbol	Size (bytes)
Point	S_p	16
MBR	S_M	32
Digest	S_h	32
RSA Signature	S_s	128

Table 1: Sizes of object attributes.

Next, we evaluate the communication cost for all methods, under kNN query processing. In particular, Figure 9 shows the total size of the $\mathcal{VO}$ as a function of k. Additionally, Table 1 summarizes the sizes (in bytes) of the various object attributes included in the $\mathcal{VO}$s of the three authentication methods. We assume that the outsourced spatial database contains a set of POIs, each with size of 16 bytes (spatial coordinates). An MBR is represented by two data points, and thus, consumes 32 bytes. Every point/MBR in the MR- and MR*-trees contains a 32-byte digest, which is computed on a leaf-level object or an MBR. Digests are computed with the SHA-256 algorithm provided in [10]. Note that we did not include the size of the tail attribute in our experiments, as it is common in all three methods. In other words, we only measure the overhead of the underlying authentication mechanisms, since tail attributes have to be transmitted to the clients even when authentication is not implemented. In the MR*-tree, all the entries of an internal or a leaf node form a KD-tree, and the digest of each entry is computed in similar fashion. When a query intersects with a node, the MR-tree returns all the non-overlapping entries to the $\mathcal{VO}$, while the MR*-tree only returns the entries that are necessary for constructing the root digest of the KD-tree. Therefore, the MR*-tree is more communication-efficient, but more computationally expensive during query processing and database updates. Figure 9 shows that VN-Auth outperforms both MR-tree variants in terms of communication cost, and it is significantly better for queries with low selectivity (i.e., for $k \leq 32$). As an example, for an 8NN query, the MR-tree returns a $\mathcal{VO}$ of 22KB while the MR*-tree returns a $\mathcal{VO}$ of 14KB. In contrast, VN-Auth only returns a $\mathcal{VO}$ of 1.3KB, including the signature. This is due to the fact that the SP only returns the database objects that belong to the result set. In other words, the size of the $\mathcal{VO}$ in our approach is linear to k.



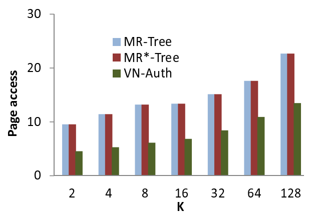
(a) CA



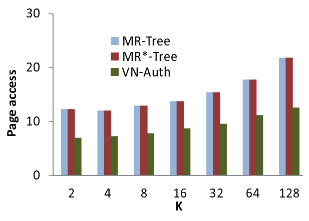
(b) NA

Figure 9: VO size vs. k.

In our next experiment, we evaluate the query processing cost at the SP in terms of I/O accesses. For VN-Auth, we perform a slight optimization on the R*-tree index, based on the neighborhood information available at the SP. Specifically, we keep the internal nodes intact, and modify the leaf nodes by storing pointers to each Voronoi neighbor of an object. The pointers facilitate easy navigation to the neighboring objects, and thus, achieve better I/O efficiency than the standard BFS algorithm. Note that such improvement is only possible for VN-Auth, as this information is incorporated in all database objects. Figure 10 depicts the I/O cost at the SP as a function of k. VN-Auth results in less I/O accesses than both MR-tree variants, in all cases. The page access cost is the same for the MR- and MR*-trees, because the MR*-tree does not store any additional information in the nodes. The idea of incorporating Voronoi diagrams into R*-trees was originally proposed in [24], where each object stored information about both its Voronoi neighbors and the corresponding cells. However, in VN-Auth, we only store Voronoi neighbors, in order to increase the fan-out of the leaf nodes.



(a) CA



(b) NA

Figure 10: Page access vs. k.

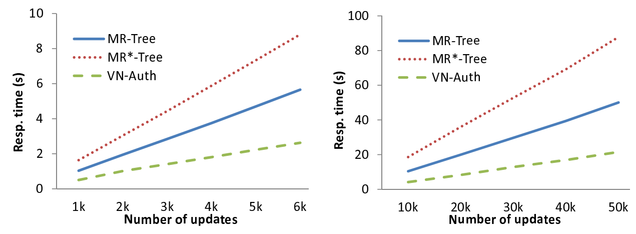
Next, we investigate the feasibility of implementing complex verification algorithms on mobile devices. As reported in [21], the cost of cryptographic primitives has become less expensive as computer hardware gets more advanced and the corresponding algorithms become more efficient. Signing an individual message (on a PC) with the RSA algorithm costs $6.06\mu s$, while verifying it takes $0.087ms$. For a condensed-RSA signature with 1000 aggregate individual signatures, the cost of signing is $0.078ms$, while the verification cost is $0.094ms$. On mobile devices, such operations are also very efficient. The costs of the cryptographic primitives in our implementation are shown in Table 2. The RSA and SHA algorithms used in our experiments are both from [10]. Signature generation/aggregation is performed on a PC, and the verification is done on an Android phone. Clearly, state-of-the-art mobile devices are more than capable of performing complex cryptographic operations, thus allowing the implementation of efficient spatial query verification techniques. Note, however, that the hashing operator is more expensive on large messages, such as the concatenation of entries (index/leaf nodes) on the MR- and MR*-trees. Therefore, computing the root digest may become more expensive than verifying a 1024-bit condensed-RSA signature on the mobile device (which is less than $5ms$, as shown in Table 2). For example, a typical 8NN query on the MR-tree returns a $\mathcal{VO}$ of 20KB. The client needs to perform the hashing algorithm 7.6 times on messages with average size of 4.5KB, which costs about $330ms$. Transferring the $\mathcal{VO}$ to the client takes an additional $300ms$. Therefore, the cost of the combined operations is much larger than the signature verification cost.

Operations	**Mobile** (ms)	**PC** (ms)
Individual signature		
Signing	–	0.02
Verifying	4.12	–
20-signature aggregation		
Signing	–	0.03
Verifying	4.48	–
Hashing (SHA-256)		
20 points/MBRs	0.44	0.04
50 points/MBRs	0.81	0.13

Table 2: Cost of primitive operations.

The next set of experiments studies the cost of database updates. Figure 11 shows the response time to complete a series of updates as a function of the number of objects being updated. Note that, for the NA dataset the total number of updates is considerably larger, in order to maintain the percentage of objects being updated similar to the CA dataset. The update cost comprises of the time for locating the object, and the time for recomputing digests and signatures as dictated by the corresponding authentication mechanisms. As discussed in Section 6, for the MR-tree based approaches, a database update affects all the nodes from the leaf to the root, because it triggers a series of hash digest recomputations in a bottom-up fashion. In VN-Auth, however, only neighboring objects are modified by the DO during an update. Consequently, VN-Auth completes the update process 2 times faster than the MR-tree, and 3 times faster than the enhanced MR*-tree approach. The time cost includes both CPU and I/O cost. We use 4KB page size and 2MB main memory for buffering tree nodes.

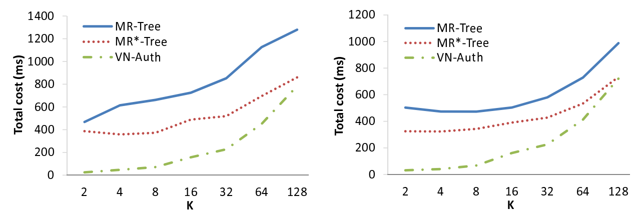
Finally, our last set of experiments investigates the query cost at the mobile clients. In particular, Figure 12 depicts the total amount of time required for receiving and verifying



(a) CA (b) NA

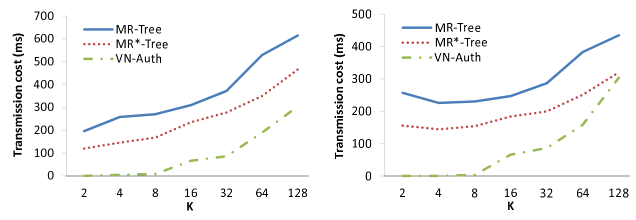
Figure 11: Response time vs. database updates.

a kNN query result as a function of k. Our experiments are conducted on Android phones that are connected to the Internet through the built-in Wi-Fi. The total query cost consists of two parts: the data transfer cost (the transmission time of the $\mathcal{VO}$), and the computational cost at the client (i.e., for result checking, digest computations, and signature verification). In other words, the query cost is measured from the moment the mobile client receives the first byte of the $\mathcal{VO}$ from the SP, until the verification process is finalized. Again, VN-Auth is superior to the MR-tree based methods, and its verification cost is significantly lower for queries that return few results. Figure 13 shows the data transfer cost for the same experiment. Comparing it with Figure 12, we conclude that the data transfer cost is a dominant factor in the overall query verification cost. Consequently, minimizing the $\mathcal{VO}$ size is an important factor in designing efficient query verification algorithms. Note that, for all three approaches, the client can determine the integrity of a query result only after it receives the complete result set from the SP. In the following section, we discuss an alternative progressive verification method that is applicable to the VN-Auth approach.



(a) CA (b) NA

Figure 12: Verification cost vs. k.



(a) CA (b) NA

Figure 13: Data transfer cost vs. k.

8. DISCUSSION

For wireless applications, due to limitations such as bandwidth constraints, unstable connections, and restricted power supply, the amount of data communicated between client and server should be kept as small as possible. Therefore, the size of the $\mathcal{VO}$ plays an important role in designing an appropriate query verification algorithm for outsourced databases. kNN queries from mobile clients usually have low selectivity (i.e., the user is interested in very few results) but require fast response time. VN-Auth is a novel approach that successfully meets these requirements in the location-based services model. However, we are aware of the fact that, when the query selectivity is high, the Merkle Hash Tree (MHT) based approaches are more efficient. In the extreme case where the client retrieves the whole database, the MHT approach would only return the root signature, while VN-Auth would need to return $6 \cdot n$ neighbors, where n is the database cardinality. Consequently, it is important to first identify the query types and user profiles in the system, and then choose a solution that accommodates the most important requirements of the application. For location-based services on mobile devices, VN-Auth is clearly a better solution.

Another important issue is providing the user with the ability to verify the query results in a progressive fashion. This is very useful in cases where decision making is based on the first few objects in the result. To ensure query integrity, the application program needs to verify the overall result before showing any part of it to the end user. For MHT based approaches, result reporting is blocked until the root digest is computed and the signature is verified. Therefore, it is not possible to pre-qualify any object in the result set, due to the hierarchical structure of the authenticated index. However, progressive result reporting is feasible under the VN-Auth framework. The result can be returned in batches, each of which can be verified independently, based on information that arrived before and not on information that is expected to arrive later. For example, the SP can divide the result set of a 20NN query into 4 batches with 5 objects each. The first 5NNs in the first batch contain the aggregate signature of the five objects. After the first batch is verified, the five objects can be reported to the end user before the verification of the second batch starts. The batches that arrive later depend on preceding ones, but not vise versa. Therefore, query verification can be performed in an incremental fashion, and results can be shown to the end user progressively. In addition, the application may allow the user to examine each batch before receiving the next one. In this way, if the user is satisfied with the current result set and does not wish to retrieve more objects, query processing may be terminated early.

9. CONCLUSIONS

In this paper, we introduced the VN-Auth query integrity assurance framework for outsourced spatial databases. Our approach separates the authentication information from the spatial index, thus allowing efficient query processing at the service provider. Additionally, since the verification information depends only on the object and its Voronoi neighbors, database updates can be disseminated quickly to their local regions, and be performed independently of all other updates in the database. VN-Auth also produces compact verification objects, which enables fast query verification on mobile devices with limited capabilities. Finally, we showed that our approach facilitates progressive result verification, which allows a user to retrieve objects in an incremental fashion, until the results are deemed satisfactory. Our simulation experiments with real-world datasets confirm that, compared to the MR-tree variants, VN-Auth produces significantly smaller verification objects, and incurs lower query verification and data update costs, especially for queries with low selectivity. In our future work, we plan to extend our methods to handle other important spatial query types, such as reverse kNN, aggregate kNN, and spatial skyline queries.

10. ACKNOWLEDGEMENTS

This research has been funded in part by NSF grant CNS-0831505 (CyberTrust), IIS-0845262 (CAREER), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), the NSF Integrated Media Systems Center (IMSC), unrestricted cash and equipment gift from Google, 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 National Science Foundation.

11. REFERENCES

[1] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. *J. Cryptology*, 17(4):297–319, 2004.

[2] W. Cheng, H. Pang, and K.-L. Tan. Authenticating Multi-dimensional Query Results in Data Publishing. In *DBSec*, pages 60–73, 2006.

[3] W. Cheng and K.-L. Tan. Authenticating kNN Query Results in Data Publishing. In *Secure Data Management*, pages 47–63, 2007.

[4] W. Cheng and K.-L. Tan. Query assurance verification for outsourced multi-dimensional databases. *Journal of Computer Security*, 17(1):101–126, 2009.

[5] A. Fiat. Batch RSA. *J. Cryptology*, 10(2):75–88, 1997.

[6] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD Conference*, pages 47–57, 1984.

[7] 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.

[8] H. Hacigümüs, S. Mehrotra, and B. R. Iyer. Providing Database as a Service. In *ICDE*, page 29, 2002.

[9] G. R. Hjaltason and H. Samet. Distance Browsing in Spatial Databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.

[10] Java SE Security. `http://java.sun.com/javase/technologies/security`.

[11] M. R. Kolahdouzan and C. Shahabi. Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases. In *VLDB*, pages 840–851, 2004.

[12] W.-S. Ku, L. Hu, C. Shahabi, and H. Wang. Query integrity assurance of location-based services accessing outsourced spatial databases. In *SSTD*, pages 80–97, 2009.

[13] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin. Dynamic Authenticated Index Structures for Outsourced Databases. In *SIGMOD Conference*, pages 121–132, 2006.

[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] E. Mykletun, M. Narasimha, and G. Tsudik. Signature Bouquets: Immutability for Aggregated/Condensed Signatures. In *ESORICS*, pages 160–176, 2004.

[17] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and Integrity in Outsourced Databases. *TOS*, 2(2):107–138, 2006.

[18] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Probability and Statistics. Wiley, NYC, 2nd edition, 2000.

[19] 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.

[20] H. Pang and K.-L. Tan. Authenticating Query Results in Edge Computing. In *ICDE*, pages 560–571, 2004.

[21] H. Pang, J. Zhang, and K. Mouratidis. Scalable Verification for Outsourced Dynamic Databases. *PVLDB*, 2(1):802–813, 2009.

[22] S. Papadopoulos, Y. Yang, S. Bakiras, and D. Papadias. Continuous Spatial Authentication. In *SSTD*, pages 62–79, 2009.

[23] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest Neighbor Queries. In *SIGMOD Conference*, pages 71–79, 1995.

[24] M. Sharifzadeh and C. Shahabi. VoR-tree: R-trees with Voronoi Diagrams for Efficient Processing of Spatial Nearest Neighbor Queries. *PVLDB*, 3(1), 2010.

[25] R. Sion. Query Execution Assurance for Outsourced Databases. In *VLDB*, pages 601–612, 2005.

[26] U.S. Census Bureau. `http://www.census.gov/geo/www/tiger/`.

[27] H. Wang, J. Yin, C.-S. Perng, and P. S. Yu. Dual Encryption for Query Integrity Assurance. In *CIKM*, pages 863–872, 2008.

[28] M. Xie, H. Wang, J. Yin, and X. Meng. Integrity Auditing of Outsourced Data. In *VLDB*, pages 782–793, 2007.

[29] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios. Spatial Outsourcing for Location-based Services. In *ICDE*, pages 1082–1091, 2008.

[30] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios. Authenticated Indexing for Outsourced Spatial Databases. *VLDB J.*, 18(3):631–648, 2009.