Location Authentication using Voronoi Diagram

Mr. Santosh B
M.Tech Student, Department of CSE
AMC Engineering College,
Bangalore, India

Mrs. Srividhya V R
Asst Prof. Department of CSE
AMC Engineering College,
Bangalore, India

***Abstract*— Cloud computing involves organizing groups of remote servers and software networks that permit centralized data storage and online access to computer services or resources. Clouds can be classified as public, private or hybrid. Outsourcing databases to the Cloud is becoming increasingly popular and has received considerable attention from the research community. In this paper, we studied the query verification problem for k-nearest-neighbor queries on road networks. While existing approaches proposed in this domain cannot verify both the distance and the shortest path to the k Nearest Neighbor (kNN) results simultaneously, we present a network Voronoi diagram-based verification approach that exploits the network Voronoi cell of each result object to verify the precision and completeness of the kNN result with regard to both distance and path. In addition to the basic verification algorithm (NVD), an improved version (DIST) with pre-computed distances within each network Voronoi cell is presented for kNN query verification to further reduce the verification cost on mobile clients. Our experiments on real-world road networks show that both verification schemes generate compact verification objects and allow for efficient verification processes on modern mobile devices.**

Index Terms*— Cloud computing; Outscourcing; k Nearest neighbor; Voronoi; pre-computed;

I. INTRODUCTION

The combination of mobile devices and Cloud-based solutions is creating a versatile ecosystem for reshaping the way geospatial data are stored, managed, served, and shared. In this new ecosystem, also known as n-database outsourcing, the data owner (DO) delegates the management of its database to a third-party Cloud service provider (SP), and the SP server is responsible for indexing the data, answering client queries, and updating the data on requests from the DOs. Mobile clients, which are used to send their queries to DOs, now submit queries to SP and retrieve results from SP directly.

Cloud computing involves organizing groups of remote servers and software networks that allow centralized data storage and online access to computer services or resources. Clouds can be classified as public, private or hybrid. And also focuses on maximizing the effectiveness of the shared resources. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand. Cloud computing provides flexible resources that can easily be scaled up or down based on user demands, effectively reducing the operational and maintenance expenses for data owners. The new Pay-As-You-Go (PAYG) business model and Data as a Service (DaaS) model have shown to be good fits for businesses with dynamic requirements and needs.

PAYG permits a user to, customize scale and provision computing resources, including storage, software and development platforms. Resource charges are based on used services, versus an entire infrastructure. DaaS brings the view that data quality can happen in a centralized place, cleansing and enriching data and offering it to different systems, applications or users, irrespective of where they were in the organization or on the network. DaaS solutions provide the following advantages: Agility, Cost-effectiveness and Data quality. Software as a Service (SaaS) works similarly, where a user leases software and customized features. Storage as a Service (SaaS) billing rotates on a regular basis because storage requirements increase are usually subject to regularly increased pricing.

Outsourcing databases to the Cloud is becoming increasingly popular and has received considerable attention from the research community. Although database outsourcing provides data owners with a more efficient, economical, and flexible solution, it also introduces new concerns [2]. with the growing popularity of the Cloud, more and more security breaches and attacks on such systems have been brought to people's attention. For example, in June 2011,Dropbox temporarily allowed visitors to login to any of its25 million user accounts due to a software glitch [4]. As a result, although commercial SPs are generally unlikely to provide inaccurate results to users, they could act involuntarily malicious and serve false results unintentionally to end users, e.g., when the SP server or the communication channel between SP and clients is compromised by attackers. Therefore, providing a mechanism that allows clients to authenticate the correctness and completeness of the query result is necessary. Specifically, correctness means all data returned by SP originate from DO without any falsification and the query result is identical to that computed by DO. Completeness means all eligible results have been included by SP in the result set, i.e., there is no false missing of correct answers.

Hacigumuset et al., was the first to propose the idea of outsourcing databases to third-party service providers in their pioneering work [2]. Afterward, numerous query authentication solutions have been proposed for outsourced relational databases [21]–[27]. Mykletunet et al., [23] provided techniques based on digital signature aggregation to ensure

data integrity and authenticity for outsourced databases. However, the techniques cannot assure completeness of the result set. Employed 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[24].For authenticating spatial queries in the Euclidean space, [6]proposed VR-tree-based mechanisms for verifying kNN and several other spatial query types on multi-dimensional databases. Their mechanisms ensure that query results are complete, authentic, and minimal.

Most existing works [6]–[14] solve query verification problems in the Euclidean space where the distance between two objects is measured by a straight line. That is, the distance depends only on the locations of the two points. Unfortunately, Euclidean distance is not an appropriate distance measure for many real-world applications where objects can only move on predefined paths, such as streets on a road network.

Outsourcing spatial databases to the cloud delivers a flexible and economical way for data owners to deliver spatial data to users of location-based services. However, in the database outsourcing paradigm, the third-party service provider is not always trustworthy, therefore, ensuring spatial query integrity is censorious. In this paper propose an efficient road network *k*-nearest-neighbor query verification technique which uses the network Voronoi diagram and neighbors to prove the integrity of query results. We propose two update modes: the one-by-one update mode and the batch update mode. Finally, we conduct extensive experiments using real-world and synthetic datasets to evaluate the verification performance and the database update cost.

Unlike preceding work that substantiate *k*-nearest neighbor results in the Euclidean space; our approach needs to justify both the distances and the shortest paths from the query point to its *k*NN results on the road network. Most existing works solve query verification problems in the Euclidean space where the distance between two objects is measured by a straight line. That is, the distance depends only on the locations of the two points.

II. PRELIMINARY

An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed. Modules are shown in the Fig 1:

A. Mobile Clients

The mobile clients send the query to the Cloud Service Provider. We study the one of the concerns called Query integrity concern. That is, how to ensure that the query results returned by Service provider are still trustworthy. The query integrity assurance is based on digital signatures and utilizes a public-key cryptosystem, such as RSA.

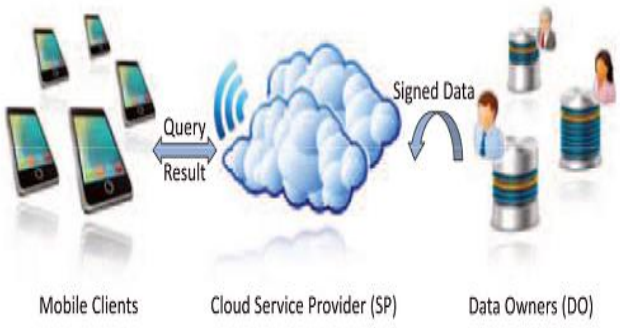

Fig 1: Database outsourcing architecture.

B. Client Service Provider

Although the Service provider is not the real owner of the data, it might return incorrect results to mobile clients out of its own interests. Moreover, a Service Provider might return suboptimal results to query clients by applying flawed or inferior algorithms in order to save computation resources. For example, as of today, the Google Maps online service still provides users with k-nearest-neighbor results based on their Euclidean distances instead of their road network distances, not because they cannot compute the road network distances, but because it is much less expensive to compute the Euclidean distances. On the other hand, with the growing popularity of the Cloud, more and more security breaches and attacks on such systems have been brought to people's attention.

C. Data Owners

Data owners obtain a private and a public key through a trusted key distribution centre. The private key is kept secret at Data owner, whereas the public key is accessible by all clients. Using its private key, the data owner digitally signs the data by generating a number of signatures. Then, it sends the signatures and the data to SP which constructs the necessary data structures for efficient query processing. The handling verification is done by the data owners.

We study the one of the concerns called Query integrity concern. That is, how to ensure that the query results returned by Service provider are still trustworthy. The query integrity assurance is based on digital signatures and utilizes a public-key cryptosystem, such as RSA.

III. ROAD NETWORK VORONOI DIAGRAMS

In many realistic applications, the distance between two points is measured by the road network distance instead of their Euclidean distance, assuming objects can only move along street systems. A road network system can be modelled as a weighted graph G*(*V,E,W*)* consisting of a set of vertices V ={*p*1, *p*2, . . . , *pn*} and a set of edges E ={*e*1, *e*2, . . . , *em*} connecting vertices to form a graph. W represents the cost of each edge in E, for example, the distance, the travelling time or the toll fees. Let $P \subset V$ be a set of points of interest (POI). We assume all POIs are restricted on road network segments (edges).

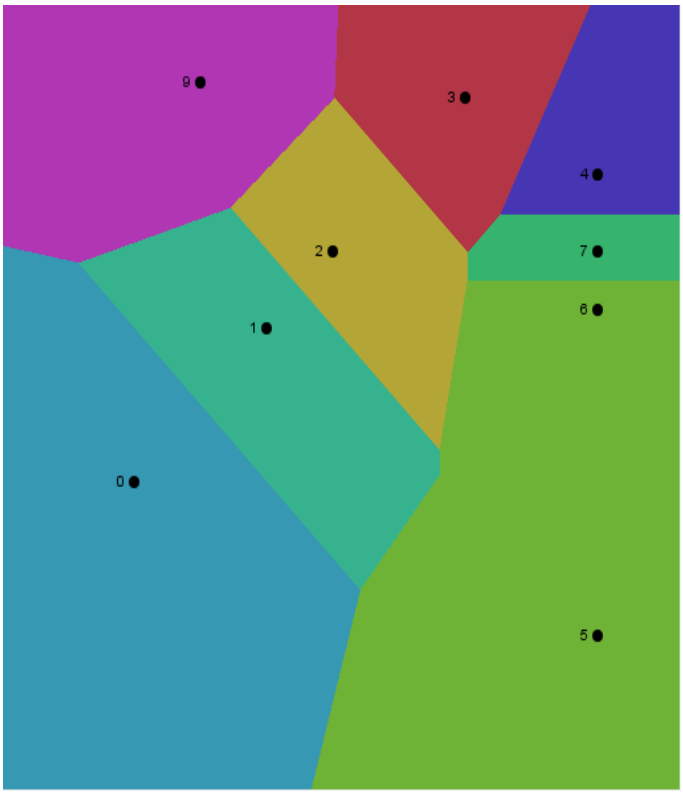

Fig 2: Voronoi diagram

IV. VERIFYING NN ON ROAD NETWORK

Nearest neighbor (NN) queries on road networks are common spatial query types for location-based services. Specifically, kNN queries enable mobile clients to retrieve the closest k POIs from the database with regard to the network distance to the location of the query point. A verifiable kNN query requests a verification object (VO) from service provider (SP) that contains not only the kNN query result, but also a proof to justify that the kNN result is correct and complete.Do not number text heads-the template will do that for you.

A. Network kNN Verification at the Client

Define abbreviations and acronyms the first time they are used in the text,

Generally, a query verification process consists of two sequential steps, that is, signature verification and geometry verification. As the signature verification step is a standard and straightforward procedure, it is only briefly discussed here and omitted from the following sections. On receiving the result of a kNN query, the client first examines the signature attached to the VO to ensure that all objects in the result set originated from DO. On receiving an aggregate signature, the client verifies the signature by employing the corresponding aggregate signature verification algorithm [23]. Signature verification can ensure the proof data has not been falsified by SP or malicious attackers.

Next, the client verifies the geometry property of the kNN result. The verification starts with verifying the first NN of the result. Recall that in the VO returned by the SP server, each POI p in the kNN result set is accompanied by its network Voronoi cell V(p) and its Voronoi neighbours Nbr(p). According to Property 5, we know that the query point q must be on a road segment inside the Voronoi cell of its first NN. Therefore, given the first NN p1 and its Voronoi cell V(p1), a client verifies that the query point q falls on one of the road segments inside V(p1). If this is true, the generator p1 is the first NN of q. Otherwise, p1 is not the first NN of q and the result is not correct. The actual distance and the shortest path from q to the generator p1 can be verified using Dijkstra's algorithm [27] or A* algorithm [28] on the sub graph inside V(p1). Note that Lemma 1 ensures the shortest path from q to p1 is fully contained in V(p1). Therefore, the Voronoi cell V(p1) is sufficient for computing the shortest path from q to p1 on the client. As a result, both the first NN p1 and the shortest path from query point q to p1 can be verified.

The network k-nearest-neighbor query verification process is shown in Algorithm 1. The inputs to the algorithm are the query point q, the verification object VO, and the parameter k. The VO contains the authenticated network Voronoi object of every POI in the kNN result, including the generator POI, the Voronoi cell of the generator and its neighbors. The method VO.getNN(i) returns the network Voronoi cell object ovc of the ith NN on line 2 and line 9. H is a min-heap containing all the Voronoi neighbors of already verified NNs and objects in H are sorted by their Euclidean distances to the query point q. Each object h in H contains a location (latitude and longitude), denoted as h.poi, the Euclidean distance to q, denoted as h.ed, and the network distance to q, represented as h.nd. The h.ed is calculated while the object is inserted into the min-heap on lines 7 and 34, and the h.nd is computed while the object is likely to be the next nearest neighbor on line 20. Visited is a set which maintains all the POIs that the client has seen so far to avoid examining the same object twice.

The algorithm starts with the first NN p1 by checking whether the query point q belongs to the road segments inside the Voronoi cell V(p1) of p1 (lines 3-5). If not, p1 is not the first NN and the verification process fails. Otherwise, p1 is verified as the first NN and is added to the Visited set on line 6. Next, the road segments inside the V(p1) are merged with g which is a sub graph of the road networks inside the Voronoi cells of already retrieved NNs, and all the neighbors of p1 are inserted into H and Visited set. The subsequent for loop (lines 8-40) iterates through each object in the kNN result set obtained from VO (line 9). If the current NN p is not in H (the current NN returned is not one of the Voronoi neighbors of previously verified NNs), then p is not the i^{th}NN of q (according to Property 6), and the verification fails on line 11. Otherwise, p is one of the neighbors of already verified NNs, and we need to verify that the network distance of p is the smallest among all the Voronoi neighbors and identical to the distance in the query result returned by SP (lines 13-39).

Algorithm 1 VerifyNetwork$k$NN(q,$\mathcal{VO}$,k)

```
1. H ← ∅; Visited ← ∅; g ← ∅;
2. ⟨p, V(p), Nbrs⟩ ← VO.getNN(1);
3. if (q ∉ V(p)) then
4.    return false;{the 1st NN fails by Property 5}
5. end if
6. Visited.add(p);
7. g ← V(p); H ← Nbrs; Visited ← Nbrs;
8. for i = 2 to k do
9.    ⟨p, V(p), Nbrs⟩ ← VO.getNN(i);
10.   if p ∉ H then
11.      return false;{Property 6}
12.   end if
13.   g ← g ∪ V(p);
14.   lb ← computeSP(g, q, p); {Lemma 2}
15.   minDist ← MaxValue; minPt ← null;
16.   for all (h ∈ H) do
17.      if (h.ed > lb) then
18.         break; {apply Euclidean restriction}
19.      else
20.         h.nd = computeSP(g, q, h.poi);
21.         if (h.nd < minDist) then
22.            minDist ← h.nd;
23.            minPt ← h;
24.         end if
25.         if (minDist < lb) then
26.            return false; {minPt is closer to q than p}
27.         end if
28.      end if
29.   end for
30.   if (p == minPt.poi && p.nd == minDist) then
31.      H.remove(minPt);{the i^th NN is verified}
32.      for all (nbr ∈ Nbrs) do
33.         if (nbr ∉ Visited) then
34.            H ← H ∪ nbr; Visited ← Visited ∪ nbr;
35.         end if
36.      end for
37.   else
38.      return false;{the i^th NN is not verified}
39.   end if
40. end for
41. return true;
```

The algorithm starts with the first NN p1 by checking whether the query point q belongs to the road segments inside the Voronoi cell V(p1) of p1 (lines 3-5). If not, p1 is not the first NN and the verification process fails. Otherwise, p1 is verified as the first NN and is added to the Visited set on line 6. Next, the road segments inside the V(p1) are merged with g which is a sub graph of the road networks inside the Voronoi cells of already retrieved NNs, and all the neighbors of p1 are inserted into H and Visited set. The subsequent for loop (lines 8-40) iterates through each object in the kNN result set obtained from VO (line 9). If the current NN p is not in H (the current NN returned is not one of the Voronoi neighbors of previously verified NNs), then p is not the ithNN of q (according to Property 6), and the verification fails on line 11. Otherwise, p is one of the neighbors of already verified NNs, and we need to verify that the network distance of p is the smallest among all the Voronoi neighbors and identical to the distance in the query result returned by SP (lines 13-39).

To apply the Euclidean restriction rule to prune the verification space, we first compute the network distance and the corresponding shortest path from q to p on the sub graph g. Then, the distance is used as a lower bound (lb) in the next steps. For each object h in H, if the Euclidean distance h.ed is greater than the lower bound, then h is skipped. Otherwise, the network distance from h to q is computed and stored in h.nd (line 20). If p has the smallest network distance among all entries in H, p is verified as the ith NN. Next, the corresponding entry is removed from H (line 31) and all the unvisited neighbors of p are added to H and Visited set (lines 32-36). Otherwise, p is not the ith NN and the verification fails on line 38.

B. Pre-computation Based Verification Scheme

During the verification in Algorithm 1, the client needs to compute the shortest distance and path from the query point *q* to candidate neighbor generators in lines14 and 20 by using Dijkstra [36] or A* [37] algorithm. Unfortunately, these distance computations are usually expensive; especially as the sub graph network *g* grows large. If the client knows distances from each border point to other border points and to the generator, then the distance computation cost can be greatly reduced. Therefore, to improve the efficiency of the client verification algorithm, we adopt the network distance pre-computation approach from [17]. Specifically, for each network Voronoi cell, DO pre-computes the shortest distance and path between two types of point pairs: (i) border point-to-generator pairs from each border point to the generator, and (ii) border-to-border point pairs between all border points on a Voronoi cell.

DO pre-compute the shortest distances and paths from the nine border point's to*p*1 and all pairs between the nine border points. After that, DO stores pre-computed distances of all point pairs in a *distance table DT (p)*. In order to reduce the storage cost, DO only stores the hash value of the shortest path between each pair of points in a separate *hashed path table* denoted $\Phi(p)$.

Next, in order to support the client's utilization of these pre-computed distances for accelerating the verification, we create a new ADS called authenticated distance table o_{adt} for each POI. Instead of incorporating the Voronoi cell *V(p)* into $o_{adt}(p)$, the distance table *DT(p)* is included. Furthermore, to ensure the integrity of paths, the hashed path table $\Phi(p)$should be included into $o_{adt}(p)$. However, in order to keep the *VO* generated later by SP as compact as possible, an accumulative hash value $\varphi(p)$ is incorporated into $o_{adt}(p)$instead of $\Phi(p)$. The value of $\varphi(p)$ is computed as the module a multiplication of all hashed paths in the pre-computed table $\Phi(p)$.

$$\varphi(p) = \left[\prod_{hp_i \in \Phi(p)} hp_i \right] mod\ n$$

Where hp_i is one hashed path value in $\Phi(p)$ and n is a 32-byte modulus that keeps $\varphi(p)$ the same size as one hash digest. In addition, n needs to be a large prime number to avoid collisions [38]. Thus, the new authentication data structure is defined as follows:

$$o_{adt}(p) = (p, DT(p), \varphi(p), Nbr(p), S)$$

$$S = sign(h(p| DT(p)| \varphi(p)| Nbr(p)))$$

After computing the $o_{adt}(p)$ for each POI p, DO outsources each $o_{adt}(p)$ and $\Phi(p)$ to SP. Eventually, the outsourced database managed by SP contains every POI p along with its old ADS $o_{anvc}(p)$, its new ADS $o_{adt}(p)$, and $\Phi(p)$.

In the query answering phase, according to the query point q, SP first gets the query result, which contains $k$nearest neighbors $\{pi|1 \leq i \leq k\}$ and the shortest distance and path from q to each neighbor POI pi. Next, SP generates the VO according to the result. Instead of returning the road segments inside Voronoi cells of the kNN result, distance tables (i.e., $DT(pi)$) are included in the VO and transferred to the query client to boost the efficiency of the verification process. Note that the SP server still needs to return the road segments inside the Voronoi cell of the first NN, namely $V(p1)$, in order to perform the verification on line 3 in Algorithm 1. Furthermore, to help the client verify the shortest paths from q to the kNN result, SP first incorporates the border points into the shortest paths in the query result. Next, instead of returning the entire hashed path table $\Phi(pi)$, SP computes another accumulative hash value $\phi(pi)$ as the modular multiplication of hashed paths between those irrelevant point pairs (i.e., the paths in $\Phi(pi)$ do not appear in the shortest paths from q to k nearest neighbors).

V. EXPERIMENTAL STUDY

In this section, we evaluate the performance of the network Voronoi diagram-based k-nearest-neighbor query verification approach through experiments. Our database outsourcing framework contains two phases: the offline database transformation phase on DO, and the online phase on SP and clients. In the offline phase, DO computes the network Voronoi diagram on the POI set and the underlying road network, and it then generates a signature for each POI by incorporating the authentication information into each POI object. In the online phase, SP evaluates queries and sends results to query clients on behalf of the DO, and then the client verifies the query result.

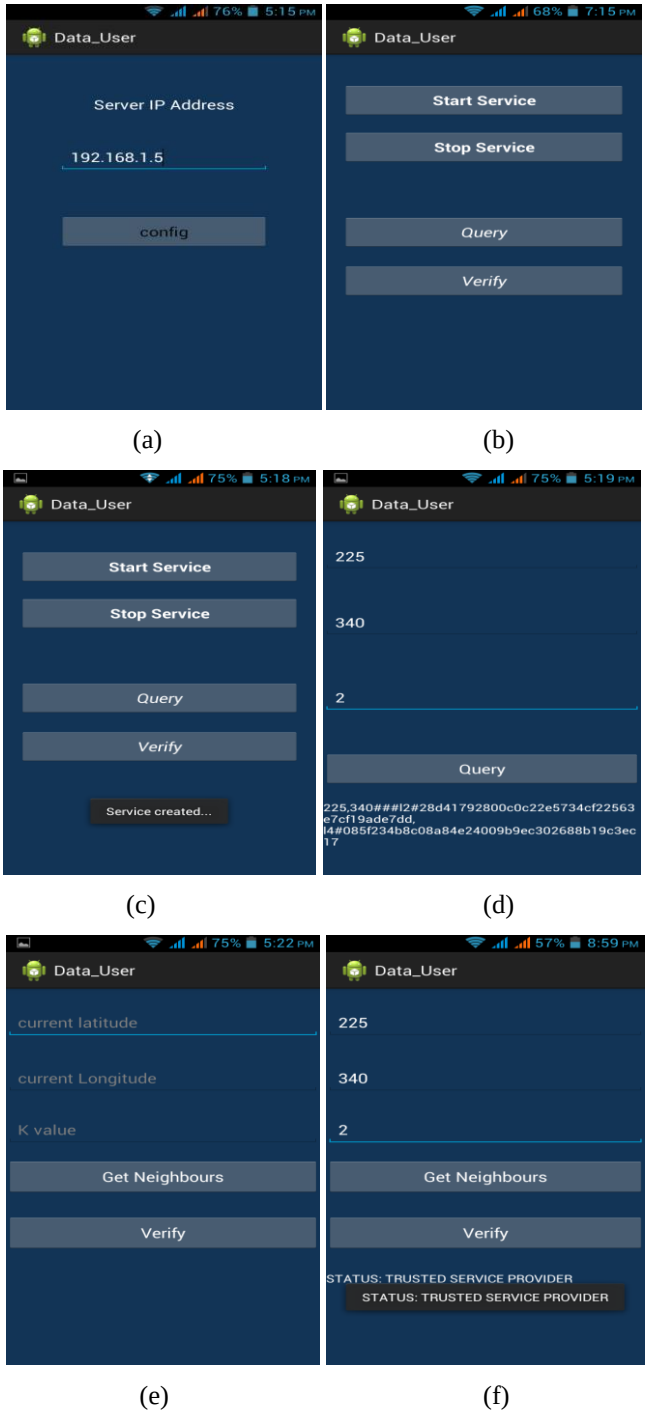

Fig 3: (a)-(f) Experimental snapshots to create service and to create trusted service provider.

Our implementation for the online query evaluation and verification is in Java. In addition, we use both real and synthetic POI datasets on these two road networks. For the CA road network, we use a real POI dataset originating from U.S. Census [41], [42] representing airports, hospitals, schools, etc. from the state of California. Additionally, to study the effect of the different density of POI datasets in our system, we experimented with a few synthetic POI datasets generated on the BAY road network. Specifically, the POIs are generated by repeatedly picking a road segment at random, and then

choosing a random point on that road segment as POI. The POI density ρ corresponds to the ratio of the number of POI points to the number of road segments in the network. As a result, the areas with a dense road network are also likely to contain dense POIs, which is consistent with our observation of the real-world, that is, more POIs are located in areas with more roads (e.g., urban areas) than areas with less roads (suburbs, national parks, etc.).

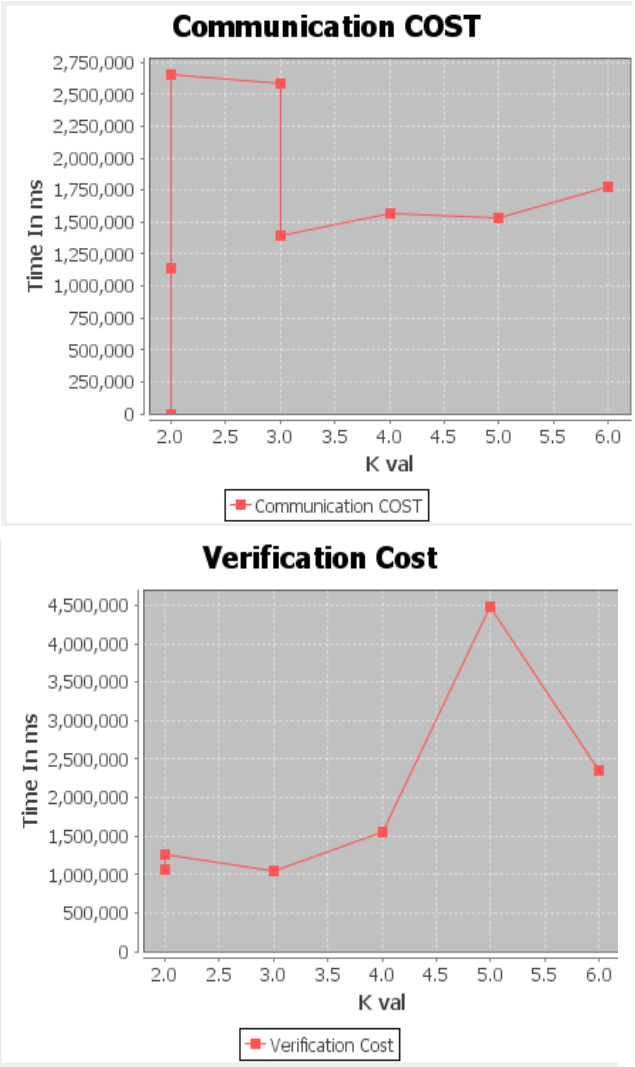

Fig 4: a) Communication cost b) Verification cost

We first study the offline database transformation costs on DO, including the total computation cost and the extra storage incurred by the authentication information. In the second set of experiments, we investigate the size of the *VO* over the query parameter *k*. Our next set of experiments studies the communication time for evaluating network *k*-nearest-neighbor queries on mobile devices. The communication time is measured by the round trip time cost per query for a client. Specifically, it starts with the client sending a query request and ends with that client receiving the result from SP, excluding the query processing time cost on SP. The next set of experiments investigates the computation time cost for verifying *k*-nearest-neighbor results on mobile devices. After receiving the *VO*, the mobile client starts the verification algorithm (Algorithm 1) to verify the correctness and completeness of the *k*NN results. Fig 3 shows some of the snapshots of experimental result to create service and to create trusted service provider. Fig 4 shows the graph of both communication cost and verification cost.

VI. CONCLUSION

Outsourcing spatial databases to the cloud delivers a flexible and economical way for data owners to deliver spatial data to users of location-based services. However, in the database outsourcing paradigm, the third-party service provider is not always trustworthy; therefore, ensuring spatial query integrity is censorious. In this paper, we studied the query verification problem for k-nearest-neighbor queries on road networks. While existing approaches proposed in this domain cannot verify both the distance and the shortest path to the kNN results simultaneously, we present a network Voronoi diagram-based verification approach that utilizes the network Voronoi cell of each result object to verify the correctness and completeness of the kNN result with regard to both distance and path. In addition to the basic verification algorithm (NVD), an improved version (DIST) with pre-computed distances within each network Voronoi cell is presented for kNN query verification to further reduce the verification cost on mobile clients. Our experiments on real-world road networks show that both verification schemes generate compact verification objects and allow for efficient verification processes on modern mobile devices.

The future work of this paper can explore a few different directions. First of all, how to handle more types of network spatial queries using the general framework and data structure discussed in this paper. Secondly, in this paper, we only considered one data owner party. However, in practice, there might be multiple data owners. For example, the POIs and the road networks can come from two different data owners. Hence, how to handle the query verification problem in the presence of multiple data owners is also an interesting direction to explore.

REFERENCES

[1] T. Brinkhoff, “A Framework for Generating Network-Based Moving Objects” GeoInformatica, 6(2):153–180, 2002.

[2] H. Hacigümüs, S.Mehrotra, and B. R. Iyer, “Providing database asa service,” in *Proc. 18th ICDE*, San Jose, CA, USA, 2002, pp. 29–38.

[3] A. Kundu and E. Bertino. Structural Signatures for Tree Data Structures.PVLDB, 1(1):138–150, 2008.

[4] A. Kundu and E. Bertino. How to Authenticate Graphs without Leaking.In EDBT, pages 609–620, 2010.

[5] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, “Spatial outsourcing for location-based services,” in *Proc. IEEE 24th ICDE*, Cancun, Mexico, 2008, pp. 1082–1091.

[6] K. Mouratidis, D. Sacharidis, and H. Pang, “Partially materialized digest scheme: An efficient verification method for outsourced databases,” *VLDB J.*, vol. 18, no. 1, pp. 363–381, 2009.

[7] W.-S. Ku, L. Hu, C. Shahabi, and H. Wang, “Query integrity assurance of location-based services accessing outsourced spatial databases,” in *Proc. 11th Int. Symp. SSTD*, Aalborg, Denmark, 2009, pp. 80–97.

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

[9] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi, "Verifying spatial queries using voronoi neighbors," in *Proc. 18th GIS*, San Jose, CA, USA, 2010, pp. 350–359.

[10] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi, "Spatial query integrity with voronoi neighbors," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 863–876, Apr. 2013.

[11] M. L. Yiu, E. Lo, and D. Yung, "Authentication of moving kNN Queries" in *Proc. IEEE 27th ICDE*, Hannover, Germany, 2011, pp. 565–576.

[12] Y. Yang, D. Papadias, S. Papadopoulos, and P. Kalnis. Authenticated Join Processing in Outsourced Databases. In SIGMOD, 2009.

[13] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. Location-basedSpatial Queries. In SIGMOD, pages 443–454, 2003.

[14] M. L. Yiu, Y. Lin, and K. Mouratidis. Efficient Verification of ShortestPath Search via Authenticated Hints. In ICDE, pages 237–248, 2010.

[15] M. R. Kolahdouzan and C. Shahabi, "Voronoi-based K nearest neighbor search for spatial network databases," in *Proc. 13th Int. Conf. VLDB*, Toronto, ON, Canada, 2004, pp. 840–851.

[16] H. Hu, D. L. Lee, and V. C. S. Lee, "Distance indexing on road networks," in *Proc. 32nd Int. Conf. VLDB*, 2006, pp. 894–905.

[17] H. Samet, J. Sankaranarayanan, and H. Alborzi, "Scalable network distance browsing in spatial databases," in *Proc. SIGMOD*, New York, NY, USA, 2008, pp. 43–54.

[18] E. W. Dijkstra, "A note on two problems in connexion with graphs," Num. Math., vol. 1, no. 1, pp. 269–271, Dec. 1959.

[19] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Trans. Syst. Sci. Cybern., vol. 4, no. 2, pp. 100–107, Jul. 1968.

[20] S. Papadopoulos, L. Wang, Y. Yang, D. Papadias, and P. Karras, *"*Authenticated Multistep Nearest Neighbor Search*,"* IEEE *Trans.data Eng.*, vol 23, no.5,pp. 641-654, May 2011.