Query Authentication of Outsourced Spatial Database

Jun Hong[1,2], Tao Wen[2], and Quan Guo[3]
[1]School of Software, North University of China, China
[2]School of Computer Science and Engineering, Northeastern University, China
[3]Computer Science and Technology, Dalian Neusoft University of Information, China

Abstract: *Outsourcing spatial database to a third party is becoming a common practice for more and more individuals and companies to save the cost of managing and maintaining database, where a data owner delegates its spatial data management tasks to a third party and grants it to provide query services. However, the third party is not full trusted. Thus, authentication information should be provided to the client for query authentication. In this paper, we introduce an efficient space authenticated data structure, called Verifiable Similarity Indexing tree (VSS-tree), to support authenticated spatial query. We build VSS-tree based on SS-tree which employs bounding sphere rather than bounding rectangle for region shape and extend it with authentication information. Based on VSS-tree, the third party finds query results and builds their corresponding verification object. The client performs query authentication using the verification object and the public key published. Finally, we evaluate the performance and validity of our algorithms, the experiment results show that VSS-tree can efficiently support spatial query and have better performance than Merkle R tree (MR-tree).*

Keywords: *Data Outsourcing, KNN, spatial database, cloud computing, query authentication.*

Received February 7, 2017; accepted December 24, 2018
https://doi.org/10.34028/iajit/17/4/12

1. Introduction

Spatial database outsourcing is motivated by a large number of practical applications, such as environmental monitoring, traffic control, geo-location services, etc. However, an obvious defect of data outsourcing is that data are remotely stored on the semi-trusted third party, which makes the data owner lose physical control of its own database. Thus, significant security problems [1, 2] are associated with data outsourcing, such as data privacy, access control, data integrity, etc. In particular, query integrity is a major issue to be resolved. This is especially important when the results are used as basis for critical decisions. Since the third party is semi-trusted to users, it may return incorrect query results. For example, a user wants to find the three nearest restaurants in Figure 1.

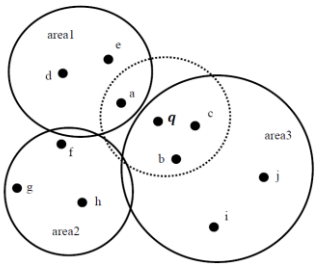

Figure 1. *kNN* query.

The real result is {a, b, c}, however, the remote server may return {b, c, f} where f is a paying user. In the worst cases, query results may include false or modified records. Therefore, the third party should provide users with authentication information for query verification. In data outsourcing, query integrity includes two aspects: correctness and completeness. Correctness means that all the records in the results do exist in data owner's original database and are not modified by anyone else. Completeness means that all the records that satisfy the query condition are in the query results. In this paper, we propose an efficient spatial authenticated structure, called Verifiable Similarity Indexing tree (VSS-tree), for spatial query authentication. Our contributions are shown as follows.

- Propose a novel spatial authenticated structure that improves query verification efficiency and reduces the size of authentication information;
- Formulate detailed cost analysis for all schemes that take into account;
- Implement VSS-tree and perform a comprehensive evaluation and comparison with Merkle R tree (MR-tree).

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 proposes system model. Section 4 details the VSS-tree. Section 5 shows the performance cost and experimental results. Finally, section 6 concludes this paper.

2. Related Work

In multi-dimensional authentication, Cheng *et al.* [4] proposed Verifiable KD tree (VKD-tree) and VR-tree, where signature chain is applied to KD tree and R tree to ensure query integrity. However, the computation of signatures resulted in a large amount of computational overhead. Yang *et al.* [22] proposed an authenticated structure, called MR-tree, for authenticating spatial queries, and an improved method, using synchronized cache, is proposed to reduce Verification Object (VO) size. MR-tree extends R* tree with authentication information as the MBT. The hash value of a leaf node is computed as h=hash ($R_1|R_2|\ldots|R_f$). An entry E in a non-leaf node is defined as E=(p_i, MBR_i, h_i), where p_i points to the ith child, MBR_i is the minimum boundary rectangle that encompasses all the regions of its ith child, h_i summarizes all the MBRs and their digests of the ith child node, i.e., $h_i=h(MBR_1|h_1|\ldots|MBR_f|h_f)$. A depth-first traversal is performed on the MR-tree for a range query. VO includes all the data entries in the visited leaf nodes and the pair of MBRs and digests in the sibling nodes pruned of the visited internal nodes.

Based on MR-tree, Yung *et al*. [17] proposed a verification scheme for query verification of moving range queries. The remote server answers the user with the query results and corresponding safe region where the query point is currently located. The user verifies the query results based on the returned safe region. Furthermore, an optimization scheme is proposed to reduce VO size, and thus reduces the network communication overhead. However, extra computation overhead is required to build the safe area. Zhang *et al*. [24] proposed a novel distributed spatial authentication data structure, called distributed MR-tree, to for query verification of k-Nearest Neighbour (kNN) queries.

Papadopoulos *et al*. [13] proposed a scheme that addressed continuous range processing and authentication on highly dynamic spatial databases, where Hilbert curve is used for transforming multiple-dimensional data into one-dimensional data and then MBT is built on the transformed data. Ku *et al*. [8] employed Hilbert curves to protect the privacy of outsourced data, and probabilistically replicated and encrypted a portion of outsourced data for query authentication. Location-based spatial query is proposed in [6, 18, 21]. Given a moving query, an answer and its safe region are returned to the client. The answer remains the same as long as the query point is in the safe region. When the query point moves out of the safe region, another answer along with its safe region and corresponding VO are returned to the client.

Hu *et al*. [5] proposed two authentication schemes based on R-tree and grid-file index for authenticating range queries. The first authenticated scheme introduced an order-insensitive method, the digest of a non-leaf node is computed as

$$h^2(dig(N.mbb)) \cdot h^2(dig(N_1)) \cdots h^2(dig(N_m)) \bmod n$$

Where *N.mbb* is the minimum bounding rectangle of node *N* itself, $N_1 \ldots N_m$ are children of *N*. Compared with MR-tree, it produced more hash computation cost. The second authenticated scheme, called grid-file index, used signature chain for query verification, however, query user has to spend more computing resources for signature verification.

Lin *et al*. [11] proposed an authentication method, called MR-Sky-tree, for location-based sky-line queries. However, it has to precompute the skyline of each data object, and thus it needs more time to build and maintain the MR-Sky-tree than the MR-tree. This method is more applicable for static or infrequently updated databases. An improved scheme of MR-Sky-tree is proposed in [10] for query verification of continuous skyline queries. Three new technologies: effective range, visible region and incremental VO, are proposed to reduce computation and communication costs.

Li *et al*. [9] proposed MKD-tree based on sliding window, which solves the integrity verification of single and continuous queries of outsourced multidimensional data streams. A spatial range query verification scheme is proposed in [3], using grid to partition spatial data, and using quad-tree to index the partitioned grid. Quad-tree index has high query verification efficiency for range queries. Zhang *et al*. [23] proposed a new authentication structure Merkle Grid and R tree (MGR-tree) for verification of spatial range queries. VO size is reduced by embedding R-tree in each leaf node of the grid tree, and thus improved the verification efficiency. Furthermore, the Hilbert curve and the filter strategy are used to build an optimal authentication index MHGR tree to speed up query verification. Jang *et al*. [7] proposed a privacy-aware query authentication scheme which guarantees data confidentiality and query result integrity for users. In this scheme, a periodic function-based data grouping scheme is designed to privately partition a spatial database into small groups for generating a signature of each group. The group signature is used to check the integrity of query results.

Polynomial Identity Random Synopses (PIRS) verification scheme is proposed in [19, 20] to support verification of grouped aggregation queries. However, PIRS can only probabilistically verify the integrity of the query results. A publicly verifiable grouped aggregation queries on outsourced data streams is proposed in [12]. The data owner defines the group in advance, and the data in the data stream is divided into different groups according to the group conditions. The data owner and the service provider maintain the aggregate value of each group incrementally. Since the group information needs to be determined in advance, the data owner and service provider have to spend a lot

of storage space to maintain the group aggregation results. Wu *et al*. [14] proposed a new authentication data structures, MIR-tree and MIR*-tree, that enable authentication of Moving top-k Spatial Keyword (MkSK) query. Xu *et al*. [16] proposed a dynamic ordered index structure, called Doit Tree, to realize integrity verification of aggregated queries for multidimensional data.

3. System Model

Figure 2 illustrates our system model. It consists of three parts: Data Owner (DO), client and Cloud Service Provider (CSP).

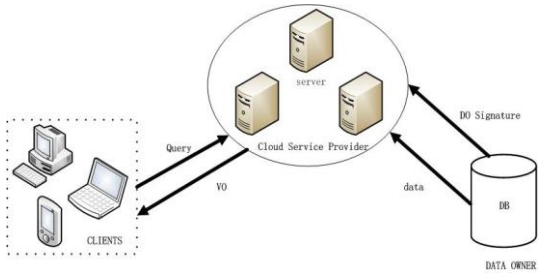

Figure 2. System model.

DO first gets a key pair from a trustful key distribution center, including a private key and a public key. Next, DO builds the authenticated spatial data structure and signs the digest of the root node to obtain signature s_{root}. Finally, DO outsources the database along with s_{root} to the CSP. CSP hosts the outsourced databases and provides query service for remote clients. CSP is semi-trusted, it performs our verification scheme honestly, but it may process the queries incorrectly. Once receiving a query, CSP traverses the authenticated structure and returns the query results along with corresponding VO for query verification. The client verifies the query results based on the VO and public key DO published. The clients only trust the public key. The main symbol list is shown in Table 1.

Table 1. Symbol list.

f_l	Fanout of leaf node	S_s	Size of a signature
f_i	Fanout of internal node	S_h	Size of a hash digest
h_{root}	Digest of root node	S_M	Size of MBR of the MR-tree
s_{root}	Signature of root node	S_C	Size of bounding sphere of the VSS-tree.
d	Dimension of data object	S_d	Size of a data object
|D|	Data cardinality	S_i	Size of an integer
H	Height of a tree	S_b	Size of the disk block
C_a	File access overhead	S_d	Size of a data object
C_h	Cost of hash operation	|p|	Size of pointer pointed to a data object
C_s	Cost of signature operation		

4. VSS-tree

SS-tree [15] is a multi-dimensional index structure designed for similarity indexing of multi-dimensional data. The structure of SS-tree is shown in Figure 3.

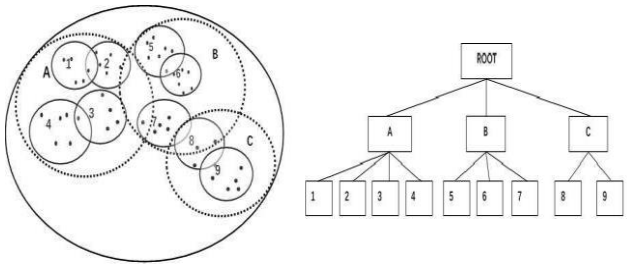

Figure 3. Structure of SS-tree.

Unlike R and R* tree, SS-tree adopts bounding sphere rather than bounding rectangle as region shape which reduces the overlap area between neighbour nodes, and thus enhances the performance on nearest neighbor queries. Another advantage of using bounding sphere is that it only spends nearly half storage compared to the bounding rectangle. Because a bounding sphere is determined by a center and a radius, the storage cost is the dimensionality of a multidimensional point plusing an integer, while a rectangle is determined by the lower and upper bound of each dimension, and the storage of a rectangle is double of dimensionality. Obviously, the degree of SS-tree is almost twice that of MR-tree, and thus reduces the height of tree.

Verifiable SS-tree (VSS-tree) is built based on SS-tree. The authenticated structure of VSS-tree is shown in Figure 4. Each entry in the internal node is associated with a digest computed by a hash function. The digest of root node, denoted as h_{root}, is signed and the signature s_{root} is published to the CSP.

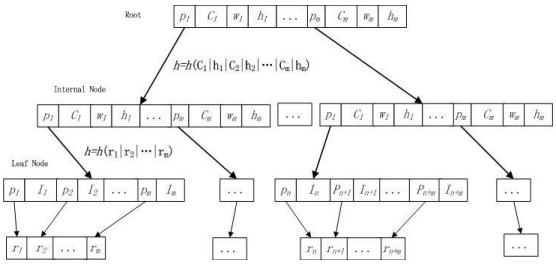

Figure 4. Structure of VSS-tree.

The structure of leaf node of VSS-tree is defined as

$$Leaf:(E_1,.,E_f)\ \ (m \le f \le M)$$
$$E_i:(p,I)\ \ (1 \le i \le f)$$

Where *m* and *M* represent the minimum and maximum number of branches of leaf nodes, respectively. Each entry in leaf nodes contains a pointer *p* and an *n*-dimensional feature vector. The structure of internal node of VSS-tree is defined as:

$$Node:(E_1,...,E_f)\ \ (m \le f \le M)$$
$$E_i:(C,p,w,h)$$

Where *C* denotes the minimum bounding sphere that encompasses all the regions of its *ith* child, *p* is a pointer that points to its *ith* child, *w* denotes the

number of points contained in the subtree whose top is the child C. The digest h summarizes the boundary spheres and digests of all its children, that is, $h=h(C_1|h_1|\ldots|C_f|h_f)$.

The center of a bounding sphere, represented as an n-dimensional point $x(x_1, x_2, .., x_f)$, is the centroid of all the data points of its children, and is computed according to Equation (1).

$$x_i = \frac{\sum_{j=1}^{f} C_j.x_i \times C_j.w}{\sum_{j=1}^{f} C_j.w} (1 \le j \le f) \quad (1)$$

Where C_j denotes the *jth* node, $C_j.x_i$ denotes the *ith* dimensional coordinate of the center of C_j, $i\,(1 \le i \le d)$ is an index of dimensions, $C_j.w$ is the number of points contained in the subtree whose top is the child C_j. The radius of a bounding sphere is computed according to Equation (2).

$$r = \max_{1 \le j \le f}(\|x - C_j.x\| + C_j.r) \quad (2)$$

Where $C_j.x$ and $C_j.r$ represent the center and radius of child C_j, respectively, $\|x-C_j.x\|$ is the distance between the center x and the center of child C_j.

4.1. Authenticated Query

4.1.1. Authenticated Range Query

In dealing with a range query Q, range query algorithm, as shown in Algorithm.1, performs a depth-first traversal on the VSS-tree to find query results and build VO.

Starting from the root node, range query algorithm visits all the entries in the internal nodes that overlap with query Q.

Algorithm 1: RangeQuery

```
Input: Node n, Query Q
Output: VO
Append [ to VO
for each entry E in n
{
  if n is a leaf node then
    Append E to VO
  else
  if dis(Q, E)≤ Q.r + E.r
    RangeQuery(E.p, Q)
  else
    append (E.C, E.h) to VO
}
Append ] to VO
```

VO includes three types of objects:

- A special token pair [and] that identify the boundaries of a node.
- All the data entries of the visited leaf nodes.
- The pair of bounding spheres and their corresponding digests of entries pruned that do not overlap with the query range.

The query client sequentially reads objects from VO to verify that:

- Each data object in VO is either a member of query results or outside query Q.
- No bounding sphere pruned in VO overlaps Q.
- The rebuilt h_{root} agrees with s_{root}.

If all the above items pass the verification, the client can confirm that the query results are correct and complete. Verification process is shown in Algorithm 2.

Algorithm 2 :Verification

```
Input: VO
Output: Circle, hash
str=null, Circle=null, Result=null
for each entry E in VO do
{
  if E is a data object
  {
    if E overlaps the query Q
      insert E into to the Result
    str = str | the binary representation of E
    recompute Circle to include E
  }
  if E is [ then
     (Cl, Hl) = Verification (VO)
  if E is a pair of Circle/Digest (Cl, Hl)
  {
    recomputed Circle to include Cl
    str=str|Cl|Hl
  }
  if e is ]
    return (Circle, hash(str))
}
```

4.1.2. Authenticated *kNN* Query

As shown in Algorithm 3, *kNN* search algorithm finds k nearest neighbours of query point q. Essentially, *kNN* search algorithm gradually increases the search radius with query point q as the center, so that the search area just contains k data points. *KnnList* contains k nearest neighbours of q, *Knn.MaxDist* represents the maximum distance between q and the data points in *KnnList*. The value of *Knn.MaxDist* is defined as $+\infty$, if *KnnList* contains less than k data objects.

Algorithm 3: kNNSearch

```
Input: n, q, k
Output: KnnList, VO
append [ to VO
if n is a leaf node
  for each entry E in n
  {
    Append E to VO
    if dist(q, E) ≤ Knn.MaxDist then
      knnList.add(E.id, distance(q, E))
  }
else
{
  for each entry E in n
    insert(BranchList, dist(q, E), E)
  sort(BranchList) by ascending
```

```
    for each entry E in BranchList
    {
        if dist(q, E) ≤ Knn.MaxDist
            kNNSearch(E.p, q, k)
        else
            append (E.C, E.h) to VO
    }
}
append ] to VO
```

During *kNN* traversal, if the currently visited node is an internal node, all its entries and their distances to *q* are inserted into the sorted list *BranchList*. Next, *kNN* search algorithm iterates through *BranchList* and recursively invokes Algorithm 3 on its visited child nodes. Once the distance of an entry to *q* is greater than *Knn.MaxDist*, the iteration will be terminated and the pairs (*Circle*,*Hash*) of the remaining entries in *BranchList* are inserted into VO. If the currently visited node is a leaf node and *KnnList* contains less than *k* data objects, the visited data is directly inserted into *KnnList*, otherwise, only the data whose distance to *q* is less than *Knn.MaxDist* is inserted into *KnnList*.

The verification process is shown in Algorithm 4, the client first computes *Knn.MaxDist*, and then sequentially read objects from VO to verify that:

- The distances between the data points not in the results and *q* are greater than *Knn.MaxDist*.
- The distances between the bounding spheres pruned and *q* are greater than *Knn.MaxDist*.
- The rebuilt hash h_{root} agrees with s_{root}.

Algorithm 4: kNNVerification

```
Input: VO, KnnList
Output: hash
str = null; C = null
for each entry E in VO
{
   if E is a data object
      if E.dist(q)≤Knn.MaxDist and E.id in KnnList
         continue
      else
         alarm the client
      str = str | the binary representation of E
   if E is [ then
      Hl= kNNVerification(VO, KnnList)
   if E is a pair of Circle/Digest (Cl, Hl) then
     str=str|Cl|Hl
   if e is ] then
      return (hash(str))
}
```

4.2. Dynamic Operation

VSS-tree supports dynamic operations of outsourced spatial database, including insertion, update and deletion. Update can be performed as a deletion followed by an insertion. Thus, we only focus on insertion and deletion. VSS-tree adopts the update algorithms of R*-tree. Both the minimum utilization of block and reinsert fraction of VSS-tree are set to 40%. When performing a deletion, the deletion algorithm first locates the leaf node N_l that contains the data object to be deleted, and then deletes it from N_l. If N_l underflows, N_l is deleted and all its rest data are reinserted to the VSS-tree. Otherwise, N_l and its affected ancestors are readjusted from bottom to top. Insertion is more complex than deletion. When inserting a new entry, the insertion algorithm locates the node *N* whose center is nearest to the new entry. If *N* is full, reinsertion will be executed, if it is still full after reinsertion, the split algorithm will be executed. In general, there are three cases and corresponding operations when inserting a new entry *E* into a node *N*:

- *N* has space, *E* is inserted there.
- *N* overflows, a part of its entries farthest from the center are deleted and reinserted into VSS-tree.
- *N* still overflows after reinsertion, split algorithm will split *N* into two nodes. The split algorithm calculates its coordinate variance to the centers of its children on each dimension and chooses the dimension with the highest variance to split it.

4.3. Proof of Query Verification

Proof of correctness: Suppose that there exists forged or modified data in query results. We know that all the data objects are involved in computing the digest h_{root}, and the digest is computed by a one-way and collision-resistant hash function. Any modification to a record makes the digest different from the original one. Furthermore, the digest of forged or altered data has to participate in the reconstruction of h_{root}, which makes the rebuilt h_{root} different from the original one, and thus disagree with s_{root}.

Proof of completeness: Let *E* in the leaf node L_n is one of the query results, but not included in the results. In order to pass the verification, CSP must make the rebuilt h_{root} match S_{root}, either *E* or the pair (Circle, hash) of L_n should be included in VO. For the first case, the verification algorithm can determine that *E* is one of the results. For the latter case, the client can detect that L_n overlaps *Q*, but is not visited by the search algorithm which violates the query verification algorithm.

5. Cost and Experiment Analysis

We first theoretically analyze the performance parameters of the VSS tree and compare them with MR-tree. Next, we conduct an exhaustive experimental evaluation to validate the effectiveness and efficiency of VSS-tree

5.1. Cost Analysis

The main performance parameters of VSS-tree considered are as follows: node fanout, index size and construction cost. Index size affects the storage cost of the member that stores the index. Index construction

cost affects the party that builds the authenticated spatial tree. If VSS-tree is built by DO and transmitted to CSP, it affects the communication cost between DO and CSP. As described in section 4, the fanout of leaf node is computed as:

$$f_l = S_b / (S_d + |p|) \quad (3)$$

The fanout of internal node is computed as:

$$f_i^{VSS} = S_b / (S_C + S_H + |p| + S_i) \quad (4)$$

The fanout of MR-tree is computed as follows.

$$f_i^{MR} = S_b / (S_M + S_h + |p|) \quad (5)$$

The fanout of VSS-tree and MR-tree under different dimensions are shown in Table 2.

Table 2. Fanout of ASDS.

Index/Dimension	3	6	12
VSS-tree	50	39	26
MR-tree	46	29	17

The height of the VSS-tree is computed as:

$$H = 1 + \log_{f_i}(|D| / f_l) \quad (6)$$

Compared with MR-tree, S_M is greater than (S_C+S_i), because the former is denoted and stored by two *n*-dimensional data, while the latter is an n-dimensional data plusing an integer. So the height of VSS-tree is lower than that of MR-tree as shown in Table 3.

Table 3. Height of ASDS.

	data size(×1000)						
Index/size	10	20	30	40	50	60	70~100
VSS(d = 6)	3	3	3	3	4	4	4
MR(d = 6)	3	3	4	4	4	4	4
VSS(d = 12)	3	4	4	4	4	4	4
MR(d = 12)	4	4	4	4	4	4	5

The storage cost of the VSS-tree is denoted as

$$S_{index} = s_b(\sum_{i=1}^{H-2} f_i^i + |D| / f_l) \quad (7)$$

The initial construction overhead of the VSS-tree is

$$C_{init}^{VSS} = C_S + C_H(\sum_{i=0}^{H-1} f_i^i) \quad (8)$$

5.2. Experiment Analysis

All experiments were performed with a Pentium Dual-Core 2.60GHz CPU and 4.0G RAM. All the programs were implemented in Java with 2Kbytes page size. Each experiment was repeated 100 times and the average was used to compare the performance of the two ASDS. The data cardinality varies from 1×10^4 to 1×10^5 and the data points are uniformly distributed in the data set. We evaluate and compare the performance parameters of MR-tree and VSS-tree from the following aspects: construction time, index size, query processing time, disk accesses, VO size and query integrity verification time.

Figure 5 illustrates construction time under different data dimensions and data cardinalities, respectively. The time includes: reading data from file, hash computation and construction of authenticated structure. The horizontal axis indicates data cardinality and the vertical axis indicates construction time.

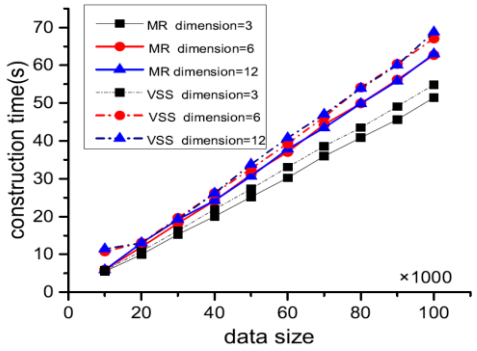

Figure 5. Construction-time vs. data cardinality and dimension.

The construction time of VSS-tree is slightly longer than that of MR-tree under the same dimension. The reason is that the computation of bounding sphere is more complex than that of MBR. A bounding sphere is represented by a centroid and a radius and computed according to Equations (1) and (2), respectively. The computation involves addition and multiplication operations, while computation of MBR only involves comparison operation.

Figure 6 illustrates the index size of two authenticated trees. The storage space of MR-tree is larger than that of VSS-tree. This is because the storage of a rectangle is double of *n*-dimension, while the storage of a bounding sphere is an *n*-dimensional point plusing an integer.

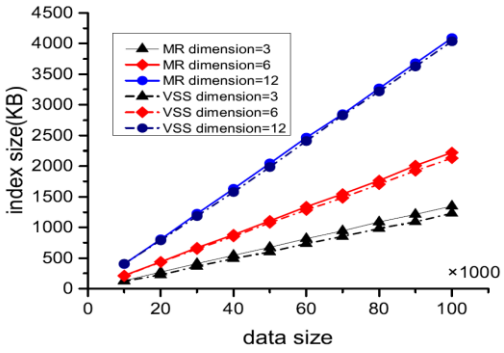

Figure 6. Index size vs. data cardinality.

Query processing cost only influences the CSP which determines the response speed of query processing. Figure 7 illustrates the query processing time of *kNN* query. Because *kNN* can be replaced by a range query whose query results just include *k* data points, the performance of range query is not discussed here. *kNN* query is executed with random query location. The value of *k* is set to 3.

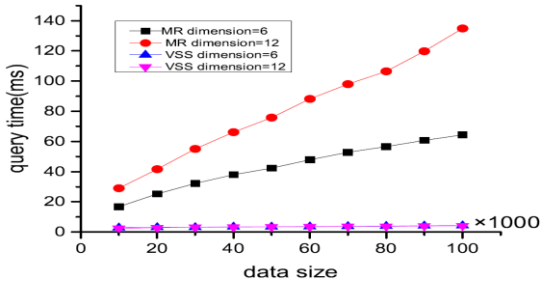

Figure 7. Query time vs. data cardinality.

The query time includes finding query results and building verification object. As can be seen from Figure 7, the query processing time based on VSS-tree is much shorter than that based on MR-tree. The reason is that using bounding sphere not only reduces the height of VSS-tree but also reduces the overlap area between nodes, and thus avoids unnecessary disk accesses.

VO size influences the network transmission overhead between CSP and clients. Figure 8 shows the VO size under different data cardinality and dimension. The VO size built based on VSS-tree is much smaller than that based on MR-tree. This is because the VSS-tree has more fanout and smaller diameter than the MR-tree under the same condition.

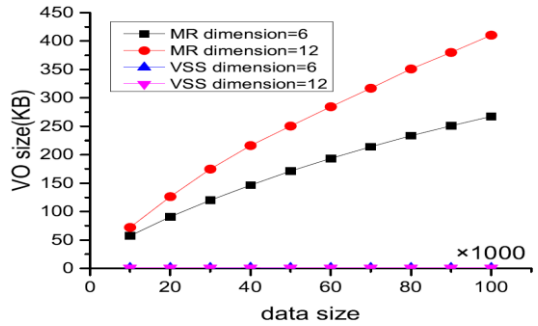

Figure 8. VO size vs. data cardinality.

Figure 9 illustrates disk accesses of *kNN* query based on VSS-tree and MR-tree. The vertical axis indicates the number of disk accesses. It can be seen that the VSS-tree needs fewer disk accesses than the MR-tree. The reason is that VSS-tree has more fanout than MR-tree which reduce the height of authenticated tree. Furthermore, VSS-tree reduces the overlap area between nodes, and thus avoids unnecessary disk accesses.

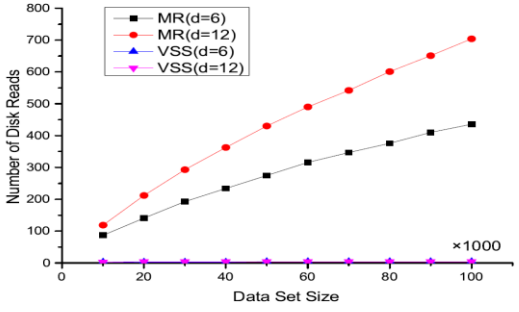

Figure 9. Number of disk reads.

Verification cost only burdens the client. Figure 10 show the verification time on the client size.

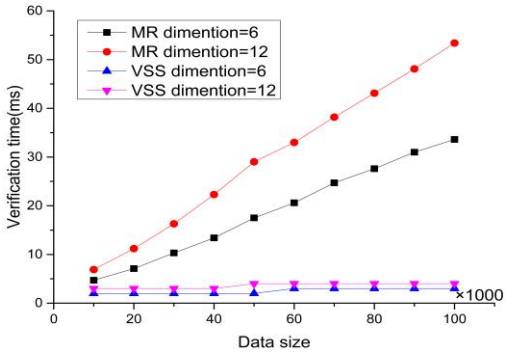

Figure 10. Verification time.

Since VO size based on VSS-tree are smaller than that based on MR-tree on the same condition, the verification time based on VSS-tree is shorter than that based on MR-tree.

6. Conclusions

In this paper, a new authenticated spatial index VSS-tree is proposed for spatial query verification. Theoretical analysis proves that VSS-tree can ensure the correctness and completeness of query results. The cost analysis and experiment results show that the query processing, query verification and network communication cost of VSS-tree is better than those of MR-tree.

In the future, we will combine query verification technique and privacy-preserving technique to perform query verification on outsourced encrypted data.

Reference

[1] Armbrust M., Fox A., Griffith R., Joseph A., Katz R, and Konwinski A., Lee G., Patternson D., Rabkin A., Syoica I., and Zharia M "A View of Cloud Computing," *Communications of The ACM*, vol. 53, no. 4, pp. 50-58, 2010.

[2] Ayyub S. and Kaushik P., "Secure Searchable Image Encryption in Cloud Using Hyper Chaos," *The International Arab Journal of Information Technology*, vol. 16, no. 2, pp. 251-259, 2019.

[3] Bo N., Xiaoxia P., Yuju L., and Xinyu P., "Query Authentications Based on A Fixed Grid Partitioning Quad-Tree Index in LBS Big Data," *Journal of Tsinghua University (Science and Technology)*, vol. 56, no. 7, pp. 785-792, 2016.

[4] Cheng W., Pang H., and Tan k., "Authenticating Multi-Dimensional Query Results in Data Publishing," *in Proceedings of IFIP Annual Conference on Data and Applications Security and Privacy*, Berlin, pp. 60-73, 2006.

[5] Hu H., Xu J., Chen Q., and Yang Z., "Authenticating Location-Based Services Without Compromising Location Privacy," *in*

Proceedings of Acm Sigmod International Conference on Management of Data, Scottsdale Arizona, pp. 301-312, 2012.

[6] Hu L., Ku W., Bakiras S., and Shahabi C., "Verifying Spatial Queries using Voronoi Neighbours," *in Proceedings of 18th SIGSPATIAL International Conference Advances in Geographic Information Systems*, California, pp. 350-359, 2010.

[7] Jang M., Yoon M., and Chang J., "A Privacy-Aware Query Authentication Index for Database Outsourcing," *in Proceedings of International Conference on Big Data and Smart Computing*, Bangkok, pp. 72-76, 2014.

[8] Ku W., Hu L., Shahabi C., and Wang H., "A Query Integrity Assurance Scheme for Accessing Outsourced Spatial Databases," *Geoinformatica*, vol. 17, no. 1, pp. 97-124, 2013.

[9] Li F., Yi K., Hadjieleftheriou M., and Kollios G., "Proof-Infused Streams: Enabling Authentication of Sliding Window Queries on Streams," *in Proceedings of 33rd International Conference on Very Large Data Bases*, Vienna, pp. 147-158, 2007.

[10] Lin X., Xu J., and Gu J., "Continuous Skyline Queries With Integrity Assurance in Outsourced Spatial Databases," *in Proceedings of International Conference on Web-Age Information Management*, Berlin, pp. 114-126, 2012.

[11] Lin X., Xu J., and Hu H., "Authentication of Location-Based Skyline Queries," *in Proceedings Of 20th ACM International Conference on Information and Knowledge Management*, Glasgow, pp. 1583-1588, 2011.

[12] Nath S. and Venkatesan R., "Publicly Verifiable Grouped Aggregation Queries on Outsourced Data Streams," *in Proceeding of IEEE 29th International Conference on Data Engineering (ICDE)*, Brisbane, pp. 517-528, 2013.

[13] Papadopoulos S., Yang Y., Bakiras S., and Papadias D., "Continuous Spatial Authentication," *in Proceedings of International Symposium on Spatial and Temporal Databases*, Berlin, pp. 62-79, 2009.

[14] Wu D., Choi B., Xu J., and Jensen C., "Authentication of Moving Top-K Spatial Keyword Queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 922-935, 2015.

[15] White D. and Jain R., "Similarity Indexing with The Ss-Tree," *in Proceedings of 12th International Conference on Data Engineering*, New Orleans, pp. 516-523, 1996.

[16] Xu J. and Chang E., "Authenticating Aggregate Range Queries Over Multidimensional Dataset," *IACR Cryptology Eprint Archive*, vol. 2010, pp. 50, 2012.

[17] Yung D., Lo E., and Yiu M., "Authentication of Moving Range Queries," *In Proceedings of 21st ACM International Conference on Information and Knowledge Management*, Maui, pp. 1372-1381, 2012.

[18] Yung D., Li Y., Lo E., and Yiu M., "Efficient Authentication of Continuously Moving Knn Queries," *IEEE Transactions on Mobile Computing*, vol. 14, no. 9, pp. 1806-1819, 2015.

[19] Yi K., Li F., Hadjieleftheriou M., Kollios G., and Srivastava D., "Randomized Synopses for Query Assurance on Data Streams," *in Proceedings of 24th International Conference on Data Engineering*, Cancún, pp. 416-425, 2008.

[20] Yi K., Li F., Cormode G., Hadjieleftheriou M., Kollios G., and Srivastava D., "Small Synopses for Group-By Query Verification on outsourced Data Streams," *ACM Transactions on Database Systems*, vol. 34, no. 3, pp. 1-42, 2009.

[21] Yiu M., Lo E., and Yung D., "Authentication of Moving Knn Queries," *in Proceeding of IEEE 27th International Conference on Data Engineering*, Hannover, pp. 565-576, 2011.

[22] Yang Y., Papadopoulos S., Papadias D., and Kollios G., "Authenticated Indexing for Outsourced Spatial Databases," *vary large Data Bases Journal*, vol. 18, no. 3, pp. 631-648, 2009.

[23] Zhang B., Dong B., and Wang H., "AuthPDB: Query Authentication for Outsourced Probabilistic Databases," *arXiv preprint arXiv*:1808.08297, 2018.

[24] Zhang C., Xu C., Xu J., and Choi B., "Distributed Knn Query Authentication," *in Proceedings of 19th IEEE International Conference on Mobile Data Management*, Aalborg, pp. 167-176, 2018.

Jun Hong received the Master degree in computer science from North University of China in 2007, China. He received his Ph.D. degree from Northeastern University in 2019, China. His current research interests are network security, cloud computing.

Tao Wen is a PhD supervisor of Northeastern University, China. He received his Ph.D. degree from Northeastern University in 1993, China. Since 2000, he has been the president of Dalian Neusoft Institute of Information, China. He has authored more than 60 refereed journals and conference papers. His research interests are network security, wireless sensor networks and service-oriented computing.

Quan Guo received his Ph.D. degree from Dalian University of Technology in 2005, China. Since 2011, he has been the vice-president of Dalian Neusoft Institute of Information, China. He has authored more than 40 refereed journals and conference papers. His research interests include network security, computer grid and optimal algorithm.