Random Sequence Coding Based Privacy Preserving Nearest Neighbor Query Method

Yunfeng Zou[1,2,3(✉)], Shiyuan Song[1,2,3], Chao Xu[1,2,3], and Haiqi Luo[1,2,3]

[1] State Grid Jiangsu Electronic Power Company Research Institute, Nanjing 210036, China
yunfeng.zou@163.com

[2] Department of Computer Science and Engineering, Southeast University, Nanjing 211189, China

[3] Nanjing Suyuan High Technology Company, Nanjing 210008, China

Abstract. Location based service has been widely used in people' life. It brings convenience to the people, in parallel with the risk of query user's location privacy disclosure. As a result, privacy preserving location based nearest neighbor queries witness its thriving in recent years. Private information retrieval (PIR) based solutions receives continuous attention for in privacy preserving for its merits in high level privacy protection strength and independence of the trusted third parties. However, existing PIR based methods fall short in high time consuming of encoding, querying efficiency and poor security to mode attacks. To address above issues, random sequence is introduced to encode POI data, which can resist mode attacks and reduce the time of data encoding. As a consequence, location privacy protection effectiveness is improved. Meanwhile, to accelerate query efficiency, a hash table structure is built at the server-side to store rules of POI distribution in the manner of space bitmap, which can position nearing POI quickly and reduce the I/O cost of database visiting efficiently. Theoretical analysis and experimental results demonstrates our solution's efficiency and effectiveness.

Keywords: Location based service · Location privacy · Private information retrieval · Random sequence

1 Introduction

The rapid development of mobile communication and spatial location technology promotes the rise of location based services (LBS) [1]. k-Nearest Neighbor (kNN) query is a query mode in location services, which refers to finding k POIs (points of interest, POI) closest to the current location of the inquirer. For example, inquirer wants to find k nearest restaurants or gas stations to his current position. The basic mode is that the inquirer submits its own location and query request to LBS server, and the server responses request and returns query result to the inquirer. During this process, the inquirer s will inevitably share their locations to an untrustable third party including the LBS service provider. There exists the risk of location privacy leakage.

Currently, existing solutions in privacy preserving location based nearest neighbor queries mainly rely on spatial confusion [2–5], data transformation [6–8], false location perturbation [9–11] and PIR (private information retrieval) technology [12–16]. Compared with the location privacy preserving technology mentioned above, PIR-based location privacy preserving querying technology can provide higher protecting strength. Based on PIR technology, inquirer can retrieve any data on an untrusted server without exposing the data information. PIR technology can compensate for the security at server side that cannot be guaranteed by traditional solutions, and provide a stronger location privacy protection. As a result, PIR technology has received continuous attention. A series of PIR-based location privacy preserving query algorithms have been proposed in recent years. Reference [14] constructs the index structure of POI dataset, and proposes a PIR based privacy retrieval algorithm. However, it spends a long time in regional division and preprocessing, and there is some difficulties to defend against pattern attacks. Reference [15] uses kd-tree, R-tree structure and Voronoi polygon to complete nearest neighbor query. However, there exists defects such as insufficient accuracy and expensive preprocessing cost. Besides, it is also not applicable to k-nearest neighbor query sceneory. Pattern attack is an important threat to location privacy protection. It means that the attacker uses the frequencies of different inquirers' database access to infer the inquirer's next target. In order to cope with pattern attacks, Reference [16] proposes an AHG (Aggregate Hilbert Grid) algorithm based on aggregated Hilbert grid, which introduces a query plan to improve query accuracy and protecting effectiveness against specific pattern attacks. However, it still has the following disadvantages:

(1) Hilbert curve-based POI encoding costs quite a lot of offline processing time at LBS server side.
(2) To cope with pattern attack, POI dataset is divided into multiple tables. In each round of the query, LBS server needs to scan the database multiple times, which leads to large processing overhead.
(3) Large number of pattern attacks facilitate attackers ability to achieve visiting frequency of different POI, which is the key of POIs' coding characteristics. Query contents can be deduced easily via leakage of coding characteristics.

To address these issues of AHG algorithm, a privacy-preserving kNN query method RSC_kNN is proposed based on random sequence and PIR technology. A random sequence is introduced to finish POI encoding, instead of the Hilbert curve. Auxiliary storage structure is built at LBS server side to reduce the cost of scanning the POI database. What's more, privacy protection effect to large amount of pattern attack is improved, as well as the query efficiency.

The main contributions of the paper are as follows:

(1) The random sequence is used instead of Hilbert curve to reduces the encoding time overhead to POI data set.
(2) A hash table structure is devised to cache some visited POIs in memory, which can reduce cost of POI database access and improve query performance.
(3) Random sequence has higher randomicity and non-repeatability, which provides higher security. Furthermore, it can be periodically updated to improve security of the system.

The paper is organized as follows: Sect. 2 summarizes related work. Section 3 describes the problem and related concepts. Section 4 introduces the algorithm of RSC_kNN. Section 5 analyzes the proposed algorithm and verifies its effectiveness. Finally, summarize the full text and look forward to the future work.

2 Related Work

2.1 PIR Technology

PIR is an important protocol proposed to protect server-side user data security. PIR protocol has the characteristics of private retrieval in the server-side database. As a result, it has become an important method to protect the location and privacy of neighbors. In order to facilitate understanding of the principle of PIR technology, the model of PIR technology is simplified as follows: suppose the database has n data blocks: $d_1, d_2, \ldots, d_n$. When the inquirer initiates a PIR protocol based data block request q(i) to the database in LBS server side, q(i) represents the i-th data block d_i in the database, the process of q(i) can be completed without any data block information leakage to potential adversary, including the LBS server. The so-called “private” means that the database server or other illegal attacker does not know the content of the user’s interest during the server query phase, thereby ensuring the security of the query initiator’s location data.

The PIR protocol can be divided into three categories: information-based security PIR, computation-based security PIR, and security-based PIR. Their security levels and application scenarios are different. Most of existing PIR based location privacy preserving methods belongs to computation-based security PIR of quadratic congruence. The quadratic congruence problem is the theoretical basis of computation-based security PIR technology. The quadratic congruence problem is a typical NP-hard problem, so it theoretically determines the security of quadratic congruence based PIR technology. Reference [8] demonstrates the security of PIR technology theoretically and proves that PIR technology can protect the location privacy of query initiators in the process of server-side data retrieval and improve system security effectively. Reference [14] builds an index structure to accelerate PIR based target POIs retrieval.

Reference [15] uses indexing structure of kd-tree and R-tree, in parallel with Voronoi polygon structure to realize POI data set division to improve query efficiency. However, voronoi polygon schema requires a large time overhead to finish POIs segmentation, and is only applicable to nearest neighbor query rather than k-nearest neighbor query mode. Reference [16] proposes the concept of "query plan" to deal with pattern attacks. However, the following problems exist: (1) The security of the algorithm depends on the encryption characteristics of the Hilbert curve and the security features of PIR protocol. In the face of large number of pattern attacks, the attacker can still calculate the encoding characteristics of POI data set, and then obtain the query content of the inquirer. (2) This algorithm encodes POI data and cooperates with a specific query plan to cope with pattern attack. However, the high complexity of the Hilbert curve in POI data encoding stage deteriorates the coding efficiency and increases the system time overhead.

2.2 AHG Algorithm

PIR based location privacy preserving technology has attracted continuous attention because of its high security and independence of trusted third parties. AHG algorithm is the most representative solution. The kNN query is implemented based on the query plan [17] by dividing POI data into multiple data tables to avoid pattern attacks and achieve strong location privacy protection.

AHG algorithm divides server-side POI data into three parts and stores them in three tables, which are DB_1, DB_2 and DB_3. DB_1 stores id of the POI point encoded by the Hilbert curve, DB_2 stores coordinates of each POI point, and DB_3 stores additional information of each POI point. The inquier initiates a kNN location based nearest neighbor query in the manner of submitting to LBS server an encrypted query request, which includes his current location Q, query content and the privacy request. After receiving the query request, LBS server searches for DB_1, DB_2 and DB_3 sequentially according to the inquirer's query request and return query results to the client in encrypted form.

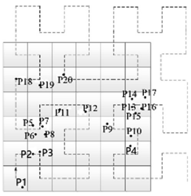
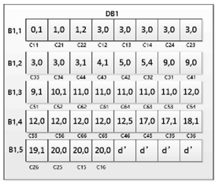


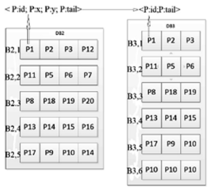
Fig. 1. POI space G encoded by Hilbert curve



DB1

B1,1	0,1	1,0	1,2	3,0	3,0	3,0	3,0	3,0
	C11	C21	C22	C12	C13	C14	C24	C23
B1,2	3,0	3,0	3,1	4,1	5,0	5,4	9,0	9,0
	C33	C34	C44	C43	C42	C32	C31	C41
B1,3	9,1	10,1	11,0	11,0	11,0	11,0	11,0	12,0
	C51	C52	C62	C61	C64	C63	C53	C54
B1,4	12,0	12,0	12,0	12,0	12,5	17,0	17,1	18,1
	C55	C56	C66	C65	C46	C45	C35	C36
B1,5	19,1	20,0	20,0	20,0	d'	d'	d'	d'
	C26	C25	C15	C16				

Fig. 2. Illustration of DB1



< P:id; P:x; P:y; P:tail> <P:id;P:tail>

DB2

B2,1	P1	P2	P3	P12
B2,2	P11	P5	P6	P7
B2.3	P8	P18	P19	P20
B2,4	P13	P14	P15	P16
B2,5	P17	P9	P10	P14

DB3

B3,1	P1	P2	P3
B3,2	P11	P5	P6
B3,3	P8	P18	P19
B3,4	P13	P14	P15
B3,5	P17	P9	P10
B3,6	P10	P10	P10

Fig. 3. Illustration of DB2, DB3

Position P is in the form of <P: id; P: x; P: y; P: tail>. P: id denotes the identifier of the position P. P: x and P: y are the coordinates of the point P. P: tail stores the additional information of P. The coordinates of point P <P: id; P: x; P: x; P: y; P: tr> and additional information of P are stored in DB_2 and DB_3, respectively, where DB_2 is associated with DB_3 via the P: tr. Figure 1 illustrates a distribution map of POI data set. In order to facilitate storage and query, the entire space is divided into a grid with scale $g \times g$ (6×6 in Fig. 1). Hilbert curve code is applied to encode the figure G. DB_1 stores the POIs' id encoded by the Hilbert curve (as shown in Fig. 2). The grid C_{14} does not contain any POI points. The Hilbert curve passes through 3 POI points, namely P_1, P_2, P_3, so $C_{14} = (3, 0)$. If the last data block does not have corresponding grid, it will be filled with a fake point. As shown in Fig. 3, DB_2 stores the coordinates of the POI point encoded by the Hilbert curve, and the name and attribute of each POI is stored in DB_3.

When inquirer initiates a k-nearest neighbor query at location Q, LBS server first finds the grid in DB_1 which is nearest to Q and obtains corresponding k ids. Subsequently, the server searches the coordinates of these ids in DB_2. Secondly, the table DB_3 is scanned to get additional information of these k POIs. Finally, k POIs' coordinates and additional information is returned to the inquirer. Due to the encryption characteristics of the Hilbert curve, AHG algorithm can improve location privacy protection strength to some extent. However it still has the following drawbacks:

(1) In the POI coding phase on the LBS server, The high complexity of Hilbert curve leads to a large time overhead in DB1 construction.
(2) POI data is divided into three tables DB1, DB2 and DB3. For each query request initiated by the client, the server needs to retrieve three tables separately, which increases the query response time.
(3) The security of the algorithm depends on the encryption characteristics of Hilbert curve and PIR. However, in the face of a large number of pattern attacks, attackers may still deduce the coding characteristics of POI datasets on the LBS server side, and then obtain specific query content, which has security risks.

3 Problem Statement and Definitions

3.1 Problem Statement

PIR based privacy preserving location-based queries needs to solve the following problems:

(1) Server-side POI data set coding can achieve good efficiency under the premise of both data availability and privacy security.
(2) Accommodating accuracy of the query, the number of client queries to search database can be reduced to improve the query efficiency.
(3) Enhance the system's ability to cope with a large number of pattern attacks.

The PIR-based AHG algorithm has many drawbacks such as long encoding time, low query efficiency and weakness in coping with pattern attack. Aiming at these problems, our solution introduces random sequence instead of Hilbert curve to encode POI dataset. A new Hash table structure is designed to store encoded POI data rather than the table DB_1 in AHG. Correspondingly, a tailed querying algorithm is proposed to complete the server-side querying.

We still follow the POI spatial graph G structure adopted in AHG algorithm. The whole data space is divided into g × g grids. As shown in Fig. 4, in the coding phase in LBS server-side, information (including the number and id of POI points) of each grid is stored in the Hash table. Meanwhile, the POI data is divided into two tables, called DB_1 and DB_2. In this stage, an efficient random sequence generation algorithm is used to generate the storage order of POI data points. POI data is stored in DB_1 and DB_2 according the storage order. As shown in Fig. 5, DB_1 stores the coordinates of POI points and DB_2 stores their additional information. When the inquirer initiates the query plan QP at location Q, LBS server responses the request and return the result to the inquirer in an encrypted form.

Random number is a sequence of randomly generated permutations. Random sequence can play the role of blur and encrypted queries, and can also rearrange and encrypt databases. A random sequence is introduced to encrypt the database and corresponding queries to make query contents indistinguishable.

Even if the number of queries is different, it can resist the pattern attack because the querier and the query content can not be accurately inferred. In addition, the generation of random sequence has the characteristics of single parameter and relatively simple process.

3.2 Definitions

Concerning these problems of Γ - privacy model, a novel privacy-preserving provenance model is devised to balance the tradeoff between privacy-preserving and utility of data provenance. The devised model applies the generalization and introduces the generalized level. Furthermore, an effective privacy-preserving provenance publishing method based on generalization is proposed to achieve the privacy security in the data provenance publishing. Relevant definitions are as follows:

Definition 1. Query Plan QP [16]: QP specifies the number of times that each data table (DB1, DB2, DB3…) needs to be retrieved for each kNN query. It depends on the kNN algorithm, size of the POI data set and the distribution of data points in the data set.

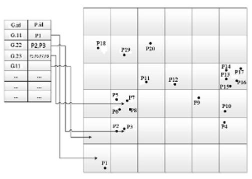


Fig. 4. Count of Hash table POI point

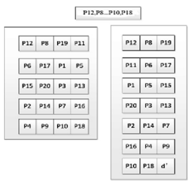


Fig. 5. POI data storage

Definition 2. POI Data Set Space Bitmap G [16]: G = <n, d, id>. N denotes the number of grids in graph G. D is the length of edges of each grid in the graph G. Symbol id is the identifier of the grid that can be uniquely distinguished from other grids after the POI data space bitmap is divided into g $\times$ g.

For example, the id of the grid in the lower left corner of Fig. 1 is G11.

Definition 3. POI Date Set S (size, array) [16]: Size denotes the number of POI points in S, and the specific information of all the points is stored in the array.

Hash table can realize fast positioning of POI data in the data query stage, reduce the number of database I/O needed in the query stage, and improve the query efficiency. Therefore, Hash table structure is designed to store POI data.

Definition 4. Hash table storage for POI data sets: The Hash table storage structure of POI dataset is as follows:

m = new Hash <G:id, <num, array>>. The key of the Hash table is used to store G: id. The value of the Hash table is used to store num and array where num and array represent number and id of the POI point, respectively.

As shown in Fig. 1, num and array of the grid G23 are <4, <5, 6, 7, 8>>, and it means that grid G23 contains four POI points P5, P6, P7, and P8.

4 Method

4.1 Algorithm RSC_KNN

In order to reduce the coding time of POI data set, random sequences is applied to encode the POI data, which can provide higher level security against large number of pattern attacks. Further, Hash table structure, in parallel with improved query strategy are leveraged to enhance query efficiency.

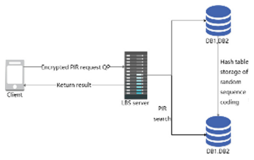


Fig. 6. Illustration of system flow

The system flow is shown in Fig. 6. The client sends the query request QP to the LBS server. At the server side, POI data is encoded by random sequence, and the encoding results are stored in the Hash table. When the server receives the client's kNN query request QP (QP contains client location information, query content and privacy requirements), it sequentially searches the hash table, DB1, DB2 to obtains k POI points including the id and the additional information, and returns the result to the client in an encrypted form.

4.2 Server-Side POI Coding

To cope with large number of pattern attacks. random sequences is introduced to encode the POI data in LBS server side. Compared with Hilbert curve, random sequence is more random and unpredictable which increases the difficulty for the attacker to obtain the client's retrieval target. At the same time, random sequence has a lower time complexity. Existing random sequence's time complexity is O(n), which is much lower than that of Hilbert curve. Data encoding work at server-side consists of two parts, to store corresponding POI information in Hash table storage and store them in the random storage.

Hash Table Constructing. The POI data set is traversed first, and the grid of the POI point is judged according to its coordinates <P: x, P: y>. The id of the Grid is used as Hash value. Hash table m is stored in system memory. It stores the number of POI points and POI point's id of each grid.

Algorithm 1 POI data set hash table storage
Input：Candidate POI point set S
Output：Hash table with POI point distribution information

```
1.  New HashMap m();
2.  For each POI in S  /* Count one POI point at a time */
3.    G:id=($\lceil \boldsymbol{P{:}\,x/G{:}\,d} \rceil$,$\lceil \boldsymbol{P{:}\,y/G{:}\,d} \rceil$);  /* Calculate the grid according to the coordinates of the POI point */
4.    m[G:id].num++;
5.    m[G:id].array.add(P:id);
6.  Return m;
```

Although the encryption feature of the Hilbert curve adopted in AHG algorithm ensures that the encoding of POI data is more concealed, Hilbert curve has high complexity when POI dataset space map G is too large. It brings large time overhead. In addition, the attacker can still find some clues related to the POI point storage order in the database by analyzing a large number of query processes and then obtain the query content of some inquirer.

In order to reduce the complexity of POI data storage and processing overhead, and improve the effect of location privacy protection, random sequence is considered to encode POI data sets. Firstly, according to the size of POI data set S, the random sequence table is generated, which consists of no duplicate values with size S: size. POI data information is stored in table DB1 and DB2 in turn. Due to the low time complexity of random sequence, the overhead of POI data coding is reduced accordingly. The security of location privacy protection can be improved further by changing the encoding mode of POI data in the way of updating random sequence and data tables DB1 and DB2, periodically.

Algorithm 2 POI data set random storage
Input：Candidate POI point set S
Output：POI points stored in database with corresponding order

```
1. set up system timer/*Update a, $DB_1$ and $DB_2$ regularly*/
2. While（timer）
3.    New array() a/* Array a is used to store the order of POI points stored in the database. */
4.    a= Random number generation algorithm（S:size）/* Use a specific random number generation algorithm to create a random number sequence*/
5.    For each POI in S
6.       Store the coordinates of all POI points in the database $DB_1$ in the order of the random sequence table a;
7.       Store the additional information of all POI points in the database $DB_2$ in the order of the random sequence table a;
```

4.3 Server-Side Query Processing

Processing POI data at the server side can improve the security of location protection and query efficiency. When the inquirer initiates a kNN query request, the server-side processing steps list as follows:

(1) The nearest grid to Q is obtained according to the location of the query point Q, and the number of POI data stored in this grid is determined according to the Hash table. If not, the nearest neighbor grid of Q is extended until the nearest neighbor grid of Q is satisfied with kNN requirement.
(2) According to the nearest neighbor grid obtained in step (1), k POI points (P: id) are retrieved from the Hash table.
(3) According to the P: id obtained in step (2), database queries satisfying the query plan are completed sequentially, and the coordinates of POI points are retrieved by DB1. According to the P: tr pointer of POI points in DB1, DB2 is retrieved sequentially to obtain additional information of POI points.
(4) returns the query result including coordinates and additional information to the inquirer in an encrypted form.

Algorithm 3 Server-side query algorithm
Input：Corrdinates of location Q, query content and query value k
Output：POI points including the coordinates and additional information

```
1.  Obtain corrdinates of location Q:Q:x,Q:y;
2.  New array() Garray;/* Garrayis used to store Q's neighboring point*/
3.  findNearestNeighborG(){/*Obtain Q's neighboring grid*/
4.  G:id=($\lceil \boldsymbol{Q}:\boldsymbol{x}/\boldsymbol{G}:\boldsymbol{d}\rceil$,$\lceil \boldsymbol{Q}:\boldsymbol{y}/\boldsymbol{G}:\boldsymbol{d}\rceil$);//Determine grid which Q belongs to according
    Q:x,Q:y;
5.  If(Q is in a single grid){
6.      Garray.add(grid);/*Add this grid to the array of Q's neighboring gird*/}
7.  If(Q is at two or more grid junctions){
8.      Garray.add(grids bordering each other);/* Add these grids to Q's neighbor grid
    array*/}
9.  Search the hash table m to obtain the total number of POI points in the neighbor-
    ing grid of Q;
10. While(count<k){
11. Extend Q's neighbor grid based on the Garray;
12.     Retrieve the hash table m and update count;}
13. New Vector() v1=Garray.P:id;/* Search the hash table m for the P:id of all POI
    points in Garray and save them in vector v1*/
14. New Vector() ret/*Save results*/
15. For each POI in v{
16.     Retrieving the coordinate information in $DB_1$ according to P: id;
17.     Ret[Pid].add(P:x,P:y); //Save the coordinate information of the point P in ret;
18.     Search the additional information of the P in $DB_2$ according to P:tr;
19.     Ret[Pid].add(P:tail);// Save the additional information of the point P in ret;}
20. Return ret;
```

4.4 Performance Analysis

This section mainly analyzes privacy protection strength and time complexity of our improved solution. In terms of location privacy preserving, our solution provides guarantee from two aspects, namely PIR and random sequence. PIR mechanism provides privacy retrieval database internally and request interface externally, which has high security strength. Random sequence reorders POI data sets to achieve double privacy protection for query process. In terms of time complexity, random sequence coding has lower time complexity than Hilbert coding, and the coding phase of POI data sets takes less time.

Compared with Hilbert curve, random sequence has the characteristics of randomness, unpredictability and non-reproducibility. At the same time, the process of retrieving database by PIR meets the basic privacy information retrieval constraints; POI data sets are also encrypted. Therefore, even through a large number of pattern attacks, attackers can not deduce the encryption rules of POI datasets, nor can they further crack PIR requests.

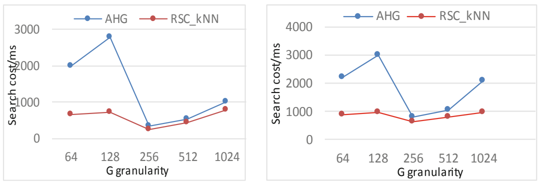
The time complexity of reordering POI data sets using random sequences in the server-side data coding stage is O (n), which is much lower than that of Hilbert curve coding O (n^2). In addition, RSC_kNN algorithm introduces the Hash table structure. The POI data information stored in DB_1 is stored in the memory through the Hash table. Each query uses only two tables, DB_1 and DB_2, which reduces the number of database I/O in the query stage, and further reduces the time cost of query processing.

In addition, the low complexity of random sequence makes the coding of server POI data set more efficient. The system can complete the re-coding of POI data set (update Hash table, DB1 table, DB2 table) by updating random sequence regularly or irregularly, thus further improving privacy protection strength.

5 Experimental Analysis

The experimental algorithm is implemented in Java and runs on Windows 10 platform with i7-3307 3.4 GHz processor and 8 GB memory. California's real POI data points were used in the experiment, including 105 K POI data points. The experiment compares our solution RSC_kNN with the AHG algorithm. The performance is mainly compared from the following aspects: G granularity selection of POI data bitmap, query time, coding time overhead of POI data set and system security in the face of pattern attack.

The grid size of POI data sets determined by the size of granularity (when POI data sets are divided into g * g grids), directly determines the number of grids, and affects the organizational structure of the database, thereby affecting the number of I/O queries to the database. Figures 7(a) and (b) show the change of query time under different granularity sizes when k = 1 and K = 5, respectively. Figure 8 describes the coding time overhead of POI data sets at system initialization.



(a) Query time（K=1） (b) Query time（k=5）

Fig. 7. Effect of granularity on query time overhead

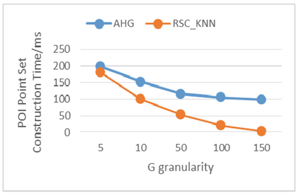
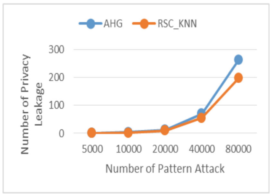


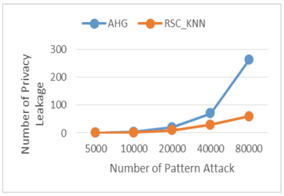
Fig. 8. Coding time overhead for POI datasets

Assuming that the number of queries Q1 and Q2 accessing the database is C1 and C2 respectively, when C1 is not equal to c2, the attacker can determine the POI of the next query or even locate the location of the query according to the frequency of query requests sent to the server by the interceptor. This attack is called pattern attack. The experiment simplifies the pattern attack, and simulates the pattern attack by predicting the POI of the next query by analyzing the frequency characteristics of query requests for DB1 and DB2 in a certain number of different queries. Figures 9(a) and (b) show the security of the query system in the face of pattern attack without updating the random sequence and updating the random sequence in time, respectively.

As can be seen from Fig. 7, the response time of RSC_kNN algorithm system based on random sequence is shorter. At the same time, the AHG algorithm is sensitive to the choice of granularity size. From Fig. 8, we can see that the RSC_kNN method based on random sequence has less time overhead and faster speed in the coding phase of POI data sets.As shown in Fig. 9(a), in the face of a large number of pattern attacks, due to the randomness and unpredictability of random sequences, the attacker infers from the existing POI data access characteristics that the possibility of querying the next POI data to be accessed by the user is lower, and the RSC_kNN algorithm based on random sequences is more secure than the AHG algorithm. At the same time, because of the short offline time of POI data encoding of RSC_kNN algorithm based on random sequence, the POI data storage of LBS server can be updated by periodically updating random sequence. From Fig. 9(b), it can be seen that under the condition of periodically updating random sequence, the system can better cope with a large number of pattern attacks, and the degree of system privacy leakage tends to be stable with the increase of attacks. Safety is higher.



(a) System security (not update random sequence)



(b) System security (update ran dom sequence periodically)

Fig. 9. System security analysis

6 Conclusion

Aiming at the shortcomings of AHG algorithm, a privacy-preserving nearest neighbor query method RSC_kNN is proposed leveraging random sequence. POI coding is realized by using random sequence. A nearest neighbor query algorithm based on auxiliary storage structure is designed on LBS server side, which can improve the intensity of location privacy protection and query processing efficiency. Theoretical analysis and experiments verify the efficiency and privacy preservation of the proposed algorithm. Effectiveness of nursing.

Whether the granularity selection of bitmap G in POI dataset is reasonable or not has a certain impact on query performance. Large or small granularity may reduce query efficiency. The optimization of storage overhead on LBS server side and the granularity setting method of POI bitmap set G will be further studied in the future.

References

1. Mokbel, M.F.: Privacy in location-based services: state-of-the-art and research directions. In: International Conference on Mobile Data Management, p. 228. IEEE Computer Society (2007)
2. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: International Conference on Mobile Systems, Applications, and Services, DBLP, pp. 31–42 (2003)
3. Bu, G.G., Liu, L.: A customizable k-Anonymity model for protecting location privacy. In: ICDCS, pp. 620–629 (2004)
4. Xiao, Z., Meng, X., Xu, J.: Quality aware privacy protection for location-based services. In: Kotagiri, R., Krishna, P.R., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 434–446. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71703-4_38
5. Gedik, B., Liu, L.: protecting location privacy with personalized k-Anonymity: architecture and algorithms. IEEE Trans. Mob. Comput. **7**(1), 1–18 (2007)
6. Wu, J., Ni, W., Zhang, S.: Generalization based privacy-preserving provenance publishing. In: Meng, X., Li, R., Wang, K., Niu, B., Wang, X., Zhao, G. (eds.) WISA 2018. LNCS, vol. 11242, pp. 287–299. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02934-0_27
7. Indyk, P., Woodruff, D.: Polylogarithmic private approximations and efficient matching. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 245–264. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_13
8. Khoshgozaran, A., Shahabi, C.: Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In: Papadias, D., Zhang, D., Kollios, G. (eds.) SSTD 2007. LNCS, vol. 4605, pp. 239–257. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73540-3_14
9. Man, L.Y., Jensen, C.S., Huang, X., et al.: SpaceTwist: managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In: IEEE, International Conference on Data Engineering, pp. 366–375. IEEE (2008)
10. Ni, W., Zhen, J., Chong, Z.: HilAnchor: location privacy protection in the presence of users' preferences. J. Comput. Sci. Technol. **27**(2), 413–427 (2012)
11. Gong, Z., Sun, G.Z., Xie, X.: Protecting privacy in location-based services using K-anonymity without cloaked region. In: Eleventh International Conference on Mobile Data Management, pp. 366–371. IEEE (2010)
12. Williams, P., Sion, R.: Usable PIR. In: Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA. DBLP (2008)
13. Yi, X., Paulet, R., Bertino, E.: Private information retrieval. J. ACM **45**(6), 965–981 (1998)
14. Khoshgozaran, A., Shahabi, C., Shirani-Mehr, H.: Enabling location privacy; moving beyond k-anonymity, cloaking and anonymizers. Knowl. Inf. Syst. **26**(3), 435–465 (2011)
15. Ghinita, G., Kalnis, P., Khoshgozaran, A., et al.: Private queries in location based services: anonymizers are not necessary. In: SIGMOD 2008, pp. 121–132 (2008)
16. Papadopoulos, S., Bakiras, S., Papadias, D.: Nearest neighbor search with strong location privacy. PVLDB **3**(1), 619–629 (2010)
17. Gedik, B., Liu, L.: Location privacy in mobile systems: a personalized anonymization model. In: Proceedings of 2005 IEEE International Conference on Distributed Computing Systems, ICDCS 2005, pp. 620–629. IEEE (2005)