International Journal of Security and Its Applications
Vol. 9, No. 5 (2015), pp. 345-360
http://dx.doi.org/10.14257/ijsia.2015.9.5.34

A Bitmap based Data Encryption Scheme in Cloud Computing

Min Yoon, Miyoung Jang, Young-Sung Shin and Jae-Woo Chang*

Dept. of Computer Engineering
Chonbuk National University
Jeonju, Republic of Korea
{myoon, brilliant, twotoma, jwchang}@jbnu.ac.kr

Abstract

With the development of cloud computing, the interest on spatial database outsourcing has been sharply increasing. Therefore, researches on data encryption methods for protecting location data privacy in outsourced databases have been actively performed. However, the existing data encryption schemes do not consider data distribution when generating data clusters and they use a tree-based index for processing k-nearest neighbor queries. As a result, the performance of query processing algorithm can fluctuate depending on the tree depth. To solve these problems, we propose a density-aware data encryption scheme and a query processing algorithm for database outsourcing. Our density-aware data encryption scheme uses a grid index to generate clusters and transforms the original data information into a bitmap. To provide efficient query processing, we use an algebraic coding based hash index to reduce the data retrieval time. Finally, in performance analysis, we show that the proposed scheme provides better query processing performance and guarantees the privacy of users, compared with the existing schemes.

Keywords: *cloud computing, location data protection, grid index, outsourced databases, density aware*

1. Introduction

Due to the recent developments of wireless positioning capabilities, such as GPS-equipped smart phones and PDA, location-based services (LBSs) have become popular. In LBS, mobile users are usually getting location information combined with traffic information, friend finder, and adjacent neighbor. The users are able to make use of these services by revealing their exact location to a location-based service provider. So, the users may meet in with a privacy violation problem. Meanwhile, due to the rapid advancements in LBSs, the speed of transmitting a terabyte of data over long distance has increased significantly and the amount of information generated in our daily lives has grown rapidly over the past decade. This large amount of information requires sophisticated management systems that are beyond the capabilities of individuals or small business. Thus, database outsourcing has been one of the most popular trends in cloud computing environment [1-9]. Because database is separated from a data owner (DO) in database outsourcing, a service provider (SP) is responsible for data storage and provides query processing for the authorized query issuer. However, because the data owner may not want to disclose the original data to the service provider, privacy issues for the database outsourcing have been studied [10-15].

For the sake of data protection in outsourced databases, there have been few encryption techniques studied. Among them, spatial transformation and distance-based data conversion techniques are widely used. The distance-based encryption method basically

* Corresponding Author

ISSN: 1738-9984 IJSIA

calculates the distance between the actual spatial data and converts their coordinates into the distance-based transformed data. The most popular distance-based encryption method, called Metric Preserving Transformation (MPT), is proposed by M. L. Yiu et al.,[16]. This method maintains the order of the original data by using the Order Preserving Encryption Scheme(OPES) [17]. Because the encrypted databases have distance oriented value, an attacker cannot infer the original coordinates from the transformed data. However, MPT suffer from the inefficient query processing since the query result includes huge number of false positive data. In addition, MPT can support only nearest neighbor (NN) queries, rather than such typical queries as range and k-NN queries. In terms of the spatial transformation techniques, M. L. Yiu. *et al.*, [16] proposed a Flexible Distance-based Hasing(FDH) that re-distributes the location space and encrypts them. This technique boots up the computation cost of data encryption by encrypting each divided area through bitmap. However, this method has some drawbacks. First, FDH technique divides the areas with anchors that are randomly selected, so that data can be skewed towards partial areas. So, if an attacker is able to find out the skewed distribution of partial area, then he/she can easily reveal real points as well. Second, the query processing time can be increased because there is a high possibility that the queries will be issued to the partial areas with skewed distribution. Third, FDH technique uses the M-tree index structure. Therefore it carries the problem that the throughput query processing performance can be increased according to the tree depth. Although M-tree was proposed as the efficient structure at the index search, but the time complexity of O(n2) is generated as the search level becomes deep.

To resolve these problems, we propose a new density-aware data encryption scheme for spatial database outsourcing. In our data encryption scheme, POIs are selected as anchor nodes to construct clusters by using a histogram. The histogram can choose anchor nodes uniformly by considering the data distribution. It means that many anchors can be chosen in dense area, while relatively less number of anchors will be chosen in sparse area. If required, our scheme re-constructs the clusters uniformly based on the a grid index to split or merge a cluster area with a given threshold. When a cluster has more POIs than the threshold, it is split into two clusters. On the other hand, when a cluster with less POIs than the threshold can be merged with its nearest cluster. Therefore an attacker cannot infer the real data distribution based on the cluster information. For efficient query processing, we propose a new query processing algorithm based on a hash index with an algebraic coding. Our contributions can be summarized as follows:

- We present a framework for providing the confidentiality of spatial data that is outsourced to cloud computing environment.

- We provide a new density-based data encryption scheme to protect original database and its distribution from an attacker and propose a GPU-based k-NN query processing algorithm for the transformed data.

- We also present an extensive experimental evaluation of our scheme by using a real data set (*e.g.*, North East USA).

The rest of this paper is organized as follows. In Section 2, we briefly review related work. In Section 3, we propose a density-aware data encryption scheme and our k-NN query processing algorithm. An empirical evaluation is presented in Section 4. Finally, Section 5 concludes our work with future research direction.

2. Related Work

For the sake of data protection from an adversary, there have been a few encryption techniques proposed in database outsourcing; distance-based, spatial-based, range-based encryption scheme.

2.1 Distance-based encryption scheme

The distance-based encryption method transforms the distance between the actual spatial data while preserving the order of the original data. The most popular distance-based encryption method is Metric Preserving Transformation (MPT) proposed by M. L. Yiu *et al.*, [16]. The method maintains the order of the encrypted data by using Order Preserving Encryption Scheme (OPES) [17]. Because this method simply calculates the distance between the spatial data, an attacker cannot infer the original coordinates from the transformed data. The MPT algorithm is processed as follows. First, a data owner chooses anchor points from the original database by using M-tree. The selected anchors form buckets with neighboring nodes. Second, all points in each bucket calculates a distance to its anchor and are applied to OPES encryption to encrypt data in ascending order. Finally, the encrypted dataset is sent to the service provider in order to process nearest neighbor queries without data decryption. The MPT scheme, which maintains the transformed distance value of data, can prevents the original coordinate of data from revealing to attackers. However, because the anchor stores distances between an anchor and data points without directions from the anchor, all data in the bucket that enclose the query point has to be returned while processing the query. This leads to significant overheads in data transmission and computation. In addition, the MPT can only support the nearest neighbor (NN) queries, but does not support range and k-NN queries.

2.2. Spatial Transformation Scheme

Spatial transformation is a process that projects the original data into another domain. First, A. Gutscher *et.al.*, [17] proposed a parallel transformation technique in which data points are transformed based on an axis. However, if an attacker has some known points with their transformed points, he/she can easily predict the transformation function. Secondly, M. L. Yiu. *et al.*, [13,14] proposed a spatial transformation technique that re-distributes the location points into the transformed space. They introduce two preliminary transformation techniques: i) Hierarchical Space Division (HSD) and ii) Error-based Transformation (ERD). Basically, HSD uses a spatial partitioning technique to redistribute the transformed data. Although this method requires a low cost for data transformation, it is vulnerable against some attack models, *e.g.*, tailored attack. For protecting against a tailored attack, they proposed ERB that utilizes a SHA-512 secure hash function for injecting noise into data. Even if ERB can protects the data against the tailored attack, it requires a high cost for data transformation. Therefore, a new enhanced HSD method (HSD*) was proposed which integrates the merits of HSD and ERB. HSD* efficiently performs a range query processing. On the other hand, they introduce a cryptographic transformation which is used for protecting data confidentiality. However, none of the existing algorithms prevents a proximity attack model that is crucial in database outsourcing. The proximity attack model means that if the approximate location of adjacent data is easy to find, an attacker can infer the location of the transformed data. In this case, the proximity attack model may cause the invasion of data privacy.

2.3. Range-based Encryption Scheme

For range-based encryption techniques, M. L. Yiu. *et al.*, [16] proposed a spatial encryption technique called flexible distance-based hashing (FDH). The FDH encrypts the original data by redistributing the location space. This technique boots up the computation cost of data encryption by transforming each divided area through a bitmap function. The algorithm of FDH is processed as follows. First, FDH randomly chooses anchor points in the original data. Secondly, it generates clusters by using the anchors with their range distances. Here, the range distances are set to be the median distances between an anchor and all points. Thirdly, FDH generates bitmap for each cluster. Fourthly, it uses M-tree [19] to support efficient query processing. In M-tree, the hamming-distance[18], which is the distance value between bitmaps of clusters, is used for retrieving the nearest cluster. For a query processing, a client encrypts a query into a bit array and finds its nearest anchor by comparing the hamming-distance between the query and anchors.

3. Density-aware Data Encryption Scheme

3.1. Background

In this paper, we assume two adversary models: honest-but-curious service provider (SP) and malicious attackers. In database outsourcing, the data owner (DO) outsources the management of his/her data to the SP. Hence, the original data to be sent to the SP should be encrypted beforehand. To protect data from the adversaries, the SP need to be able to process queries on the encrypted data. Figure 1 illustrates the architecture of our spatial database outsourcing system with three main components: a mobile user (*i.e.*, authorized user), a location-based service provider and a database owner. The overall data processing is performed as follows. In pre-processing step, the DO creates a secret encryption key, and then applies the encryption method to converts the original point set P into a transformed points set P`. Then, the transformed dataset is sent to the SP. At this time, the DO sends the encryption key to the trusted users over a secure communication channel, *e.g.*, SSL. In the query processing step, a user (U) issues a query q to SP. For this, U encodes q to q` by using the secret key from DO and sends q` to the SP. The SP evaluates q` over P` and returns the candidate results to U. Finally, U decrypts the encrypted query results and selects the genuine query results.

From the analysis of most related existing methods, FDH [16], we found the following motivations. First, FDH divides the area of the original data by randomly selecting anchors, so data can be skewed towards some anchors. Hence, if an attacker is able to find out the skewed distribution of a specific area, then he/she can easily infer the real points. Secondly, the query processing performance of FDH gets worse when the query is issued into the densely populated anchors. Thirdly, because FDH uses the M-tree index structure, it suffers from the performance degradation according to the tree depth.

In this section, we propose a density-aware data encryption and a bitmap-based hashing scheme for processing k nearest neighbor (NN) query. To resolve the problems of the existing schemes, we use a histogram to uniformly select anchor nodes for cluster construction. Then our scheme re-constructs the clusters to store the data into clusters uniformly. For this, we use a grid index to split or merge a cluster area with a given threshold. When a cluster has more points of interest(POIs) than the threshold, it is split into two clusters. On the other hand, for a cluster with less POIs than the threshold, it can be merged with its nearest cluster. Therefore an attacker cannot infer the distribution of real dataset. The main advantage of the proposed scheme is that our scheme always returns a constant-sized candidate set in one communication round.

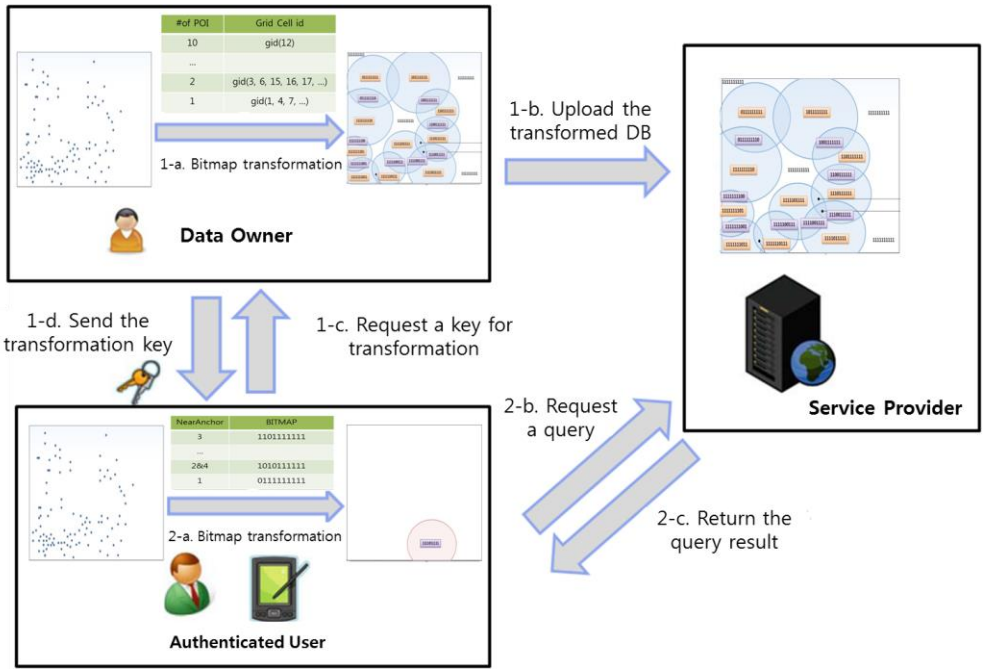


Figure 1. System Architecture

3.2. A density-based Data Encryption

Our algorithm is processed in 4 phases; anchor selection, cluster optimization, bitmap encryption, and index construction. In the anchor selection phase, we use a grid-based histogram to select anchors according to the data distribution. In the cluster optimization phase, all clusters can be adjusted for storing data into its clusters uniformly. For protecting the original data from the attacker, our algorithm transforms the clusters into a bitmap in the bitmap encryption phase. In the index construction phase, a hash-based index is constructed to support efficient kNN query processing.

3.2.1. Anchor Selection

The main problem of the existing method [16] is that the anchors can be chosen in a specific area because the existing method selects anchors randomly. If an attacker is able to find out the skewed distribution of the specific area, he/she can easily infer the real points. To solve this problem, we propose an anchor selection algorithm using a grid-based histogram to consider the data distribution. Our anchor selection algorithm is performed as follows (Figure 4). First, the original dataset P is inserted in the n x n grid index. P is represented as <id, x, y>, where id means an identifier of the original data, x and y are the coordinates of the original data. Secondly, the algorithm counts the number of data within each grid cell for making a histogram, and sorts the cells in the descending order of data counts. Thirdly, by using the histogram, the data owner computes the number of anchors based on Equation (1), and then randomly selects anchors from the cell. If a cell does not satisfy the threshold, the minimum number of data for a cluster, the algorithm expands the cell to its nearest cells to satisfy the threshold. The threshold can be given by a user or a system. The latter one is calculated by using Equation (2).

$$AnchorRatio = \frac{(\#\ of\ Anchor) \times (\#\ of\ Data\ in\ Anchor\ Range)}{\#\ of\ data\ in\ P} \quad (1)$$

$$Threshold = \frac{\#\ of\ Anchor}{\#\ of\ data\ in\ P} \quad (2)$$

For example, a grid cell 12 encloses the largest data group in Figure 2, so the anchor selection algorithm is started from the cell 12. By using Equation 1, the number of anchor is calculated as 1. Hence, one anchor is randomly selected in the grid cell 12. Because a grid cell 66 includes three data, it requires a cell expansion to satisfy the threshold. So the algorithm expands the grid cell 66 to its neighboring cells and one cluster anchor can be chosen.

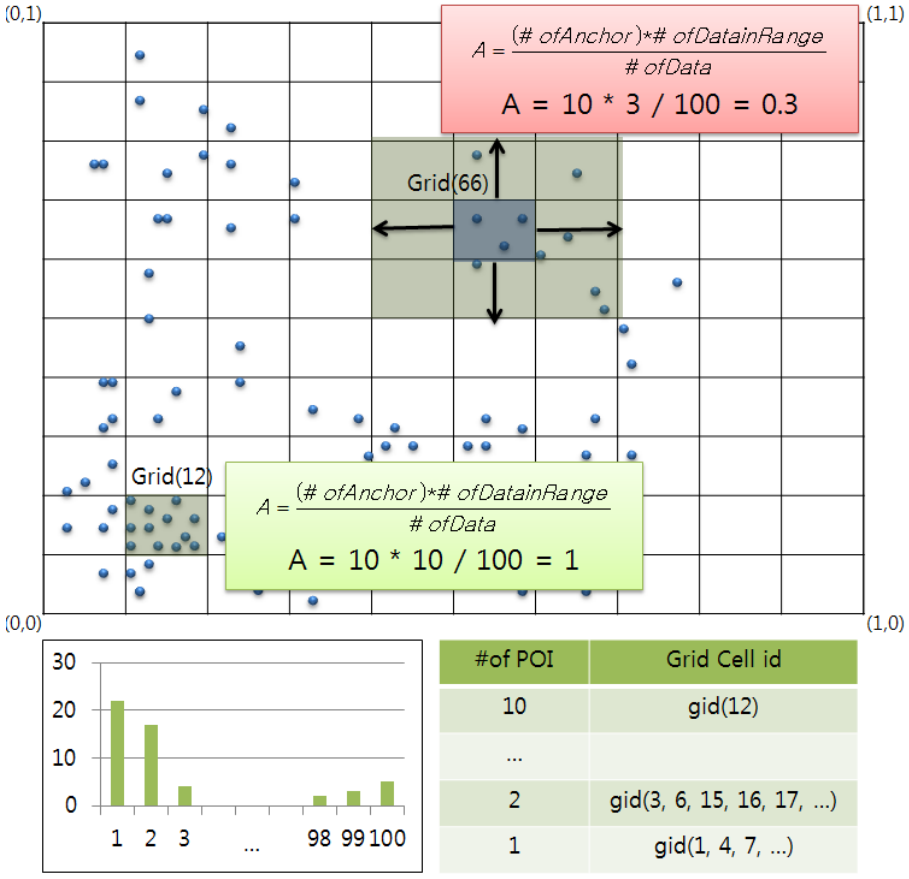
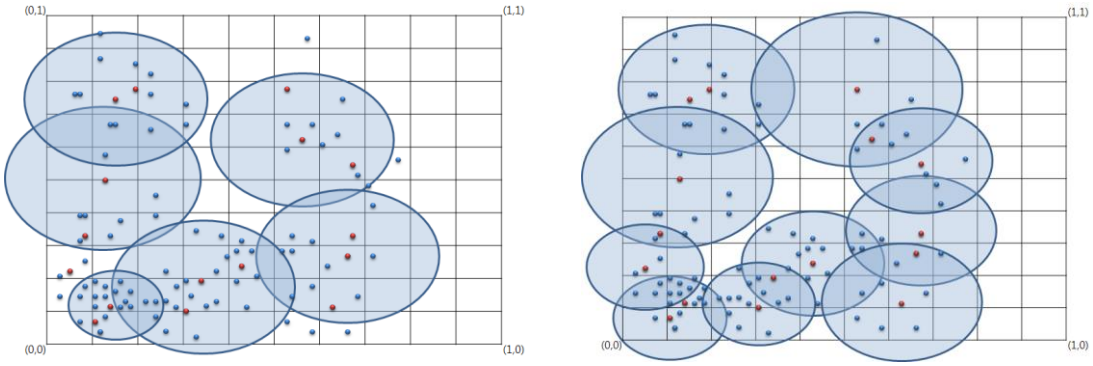


Figure 2. Histogram-based Anchor Selection

3.2.2 Cluster Optimization

Even if we perform the density-aware anchor selection phase, the data can be skewed in some clusters. To solve this problem, we propose a merge and split algorithm to store the data into clusters uniformly. Our algorithm is processed as follows. First, our algorithm calculates the merge and split thresholds of all clusters which are constructed from phase 1. Second, if the number of data in a cluster is greater than the split threshold, the algorithm divides the cluster into two groups. In order to split the cluster, the farthest point from the cluster anchor is chosen as an anchor for the new cluster. Third, if the number of the data in a cluster is lower than the merge threshold, the cluster should be merged with the nearest cluster. To merge two clusters, the midpoint between two anchors is calculated to select a new anchor. The new anchor becomes a point which is the nearest to the midpoint. The algorithm repeats until all clusters satisfy the merge and split thresholds. Figure 3 shows an

example of cluster optimization algorithm. Figure 3(a) shows the initial clusters from anchor selection phase and Figure 3(b) shows the re-constructed clusters after the cluster optimization is performed.



(a) Initial Anchor and Bound (b) Reconstructed Anchor and Bound

Figure 3. Exmaple of Anchor Bound Selection Algorithm

3.2.3 Bitmap Encryption

Let the selected anchor objects be a_1, a_2, …, a_i, a_n. For each anchor object a_i, we need to calculate a distance value r_i, which represents the cover range of the anchor. Given an object $p \in P$, we convert the coordinates of the data point into an A-length bitmap where the ith bit of the bitmap is defined in [7]:

$$Bitmap(p)[i] = \begin{cases} i = 0, if\ dist(a_i, p) \leq r \\ i = 1, otherwise \end{cases} \quad (3)$$

The resulting bitmap for an anchor is shown in Figure 4 where all data in an anchor a_i share the bitmap of a_i. The process of generating bitmap is simple and intuitive, so we omit the detailed explanation of this step.

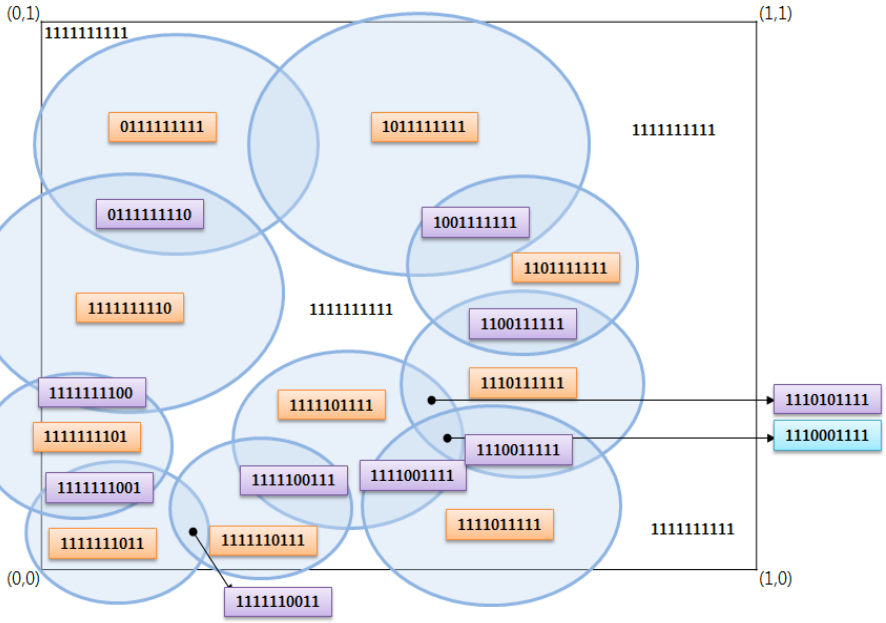


Figure 4. Example of Bitmap-Based Encryption Scheme

3.2.4. Index Construction

Once the bitmaps for all anchors are generated, we build a bitmap-based index for query processing. In order to enhance the performance of query processing, we devise a hash index based on algebraic coding for retrieving the nearest anchor bitmap rapidly. The proposed hash index uses a bit array as the coefficient of a polynomial. The address of a hash table is calculated based on the prime number that is close to the hash table size (Equation 4). We divide the bit array value by the prime number and assign its remainder as a hash table address. In equation 4, bit means an anchor bitmap and Hsize indicates a hash table size. Hence, we insert the bitmap of anchors into the hash table by applying their bitmap to the proposed hash function. The query processing performance can be improved because we can directly access the hash table at the query time.

$$hashAddress = bit\%(1 + \sum_{m-1}^{2^{Hsize}} \left[\sqrt[Hsize]{Hsize} \left(\sum_{x=1}^{n} \left[cos^2 \pi \frac{(x-1)! + 1}{x} \right] \right)^{\frac{-1}{n}} \right]) \quad (4)$$

The algorithm for the bitmap-based hash index construction is as follows. First, we calculate the hash addresses of the generated anchor bitmaps. Secondly, the anchor bitmap is inserted into the hash table. In Figure 5, we assume that the anchor bitmap is given as '10010111' and the hash table size is 21. Based on the equation 4, the hash address of this anchor is '10010' and so the anchor information is stored into the 18-th bucket of the hash table.

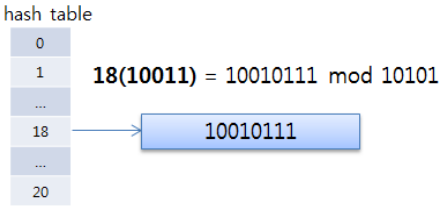


Figure 5. Example of Index Construction

3.3. Hash-based Query Processing Algorithm

This section describes our query processing algorithm to enhance the quality of service (QoS) in terms of query processing time and overheads. Figure 6 shows an overall query processing algorithm. First, a user transforms a query into a bitmap by using the given encryption key from DO. Then, the user sends the transformed query to the SP (line 1-2). Second, our algorithm retrieves distances between clusters to find the nearest cluster. If a cluster does not have enough POIs to satisfy k, then our algorithm expands the search area by selecting the next nearest anchor from the query (line 3-4) When k is satisfied, our algorithm calculates the hash address of the query and retrieves all anchors stored in the query address. Third, in order to find the nearest data point, we calculate the hamming distances [18] between the transformed query and candidates anchors (line 5-10). The Hamming distance measures the minimum number of substitutions required to change one string into the other. For binary strings a and b, the Hamming distance between them is equal to the number of ones in a XOR b. Finally, the final kNN result of the query is sent to the user and the user decrypts the result to acquire the original data coordinates (line 11-13).

Hash-based kNN Query Processing Algorithm
Input : q //query data
Output : result //nearest neighbor data
Start of Algorithm
/* in authorized client*/
1. TQ ← bitmap(q)
2. client sends TQ to service provider
/* in service provider */
3.caculate minimum distance D between clusters and TQ
4. find candidate clusters (C) in range of D
5. hash_address ← AlgebricCoding(C)
6. Access hash table using has_address
7. if(data in hash bucket > 2)
8. for each $data_i$ ∈hash bucket
9. if(candidate.dist < hammingDist($data_i$)
10. candidate.dist ← hammingDist($data_i$)
11. sends candidate to authorized client
/* in authorized client*/
12. result ← Decrypt candidate
13. return result
End of Algorithm

Figure 6. Hash-based NN Query Processing Algorithm

Figure 7 shows the example of the 1NN query processing algorithm. Assuming that a query point is (0.1, 015), the query can be transformed into '1111111001' by using the given encryption key. Based on the transformed query, the service provider accesses to the 9-th hash bucket and obtains two bitmaps '1111111001' and '1111001111'. The minimum hamming distance between the query (1111111001) and cand1 (1111111001) is calculated as zero, whereas the hamming distance between the query and cand2 (1111001111) is calculated as 4. Thus, cand1 is returned to the user as the final result.

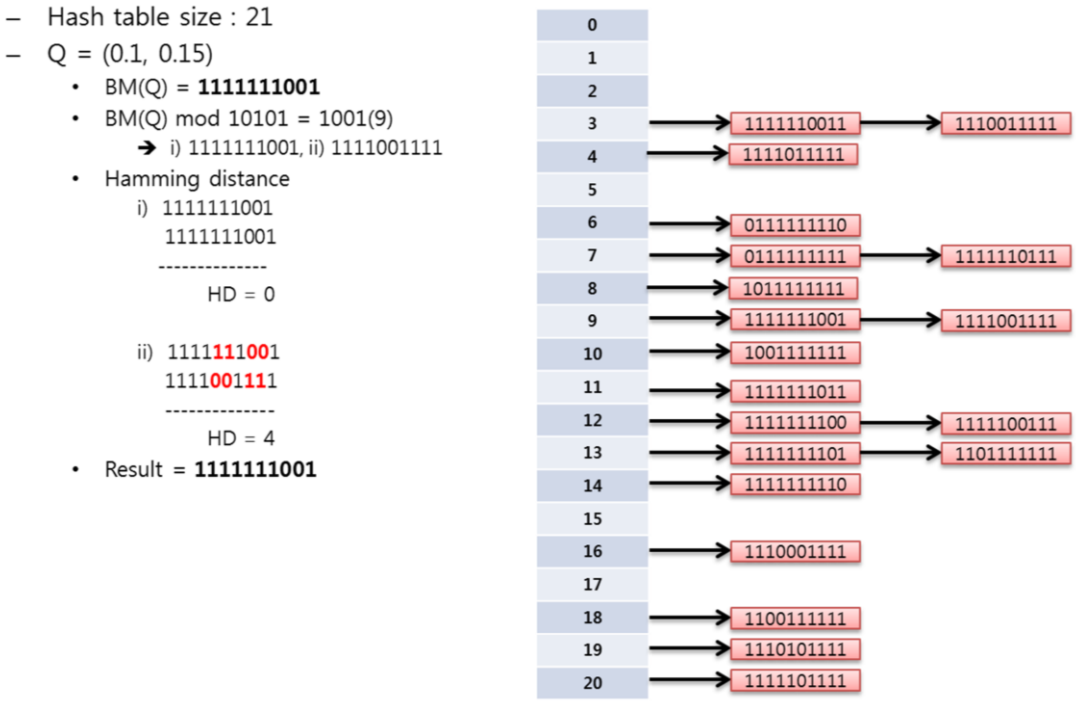


Figure 7. Example of 1NN Query Processing Algorithm

4. Performance Evaluation

In this section, we present the extensive experimental results of our encryption scheme. For this, we compare our encryption scheme with the existing FDH in terms of data encryption time, degree of data distribution, and query processing time. In our experiment, we use four spatial datasets: Uniform(100,000 points), Gaussian(100,000 points), Skewed(100,000 points) and the real dataset of Northern East America (NE) containing 119,898 point of interests (POIs), as shown in Figure 8. Table 1 represents the experimental environment.

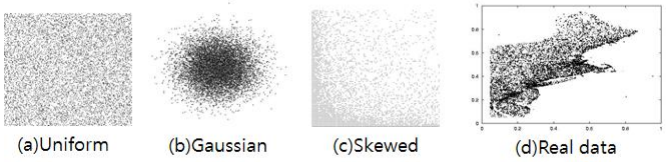


Figure 8. Data Set

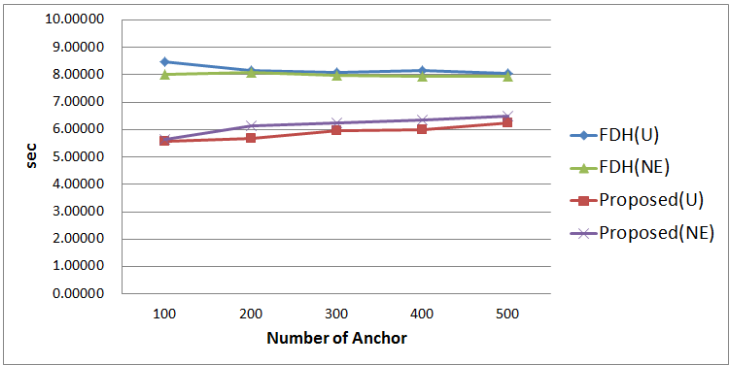
Table 1. Experimental Setup

	Performance
CPU	Intel® Core™ i3-2100 @ 3.10 GH
Memeory	4GB
O/S	Windows 7 Enterprise K
Compiler	Microsoft Visual Studio 2010

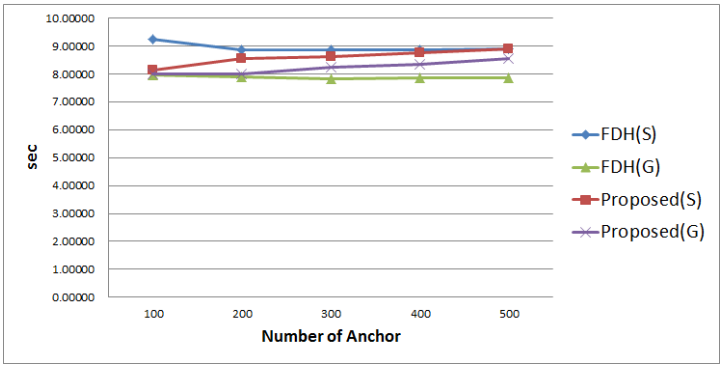
4.1. Data Encryption Time

Figure 9 shows data encryption time with varying the number of anchors. The data encryption time can be increased as the number of anchors increases. When the number of anchors is 500 in Uniform and Real data, the encryption time of our

scheme requires 6.25 seconds and 6.69 seconds respectively. On the other hand, the existing FDH scheme requires 8.31 seconds and 8.04 seconds, respectively. This is because our scheme generates clusters by using a grid index while the existing scheme uses a tree structure which has the expensive cost to build in large data. In case of Gaussian and Skewed data, the encryption time of our scheme is higher than Uniform and Real data as shown in Figure 9(b). This is because for densely populated data, merge and split operations are performed many times to store the data in a uniform way.



(a) Data Encryption Time with Uniform and Real Data

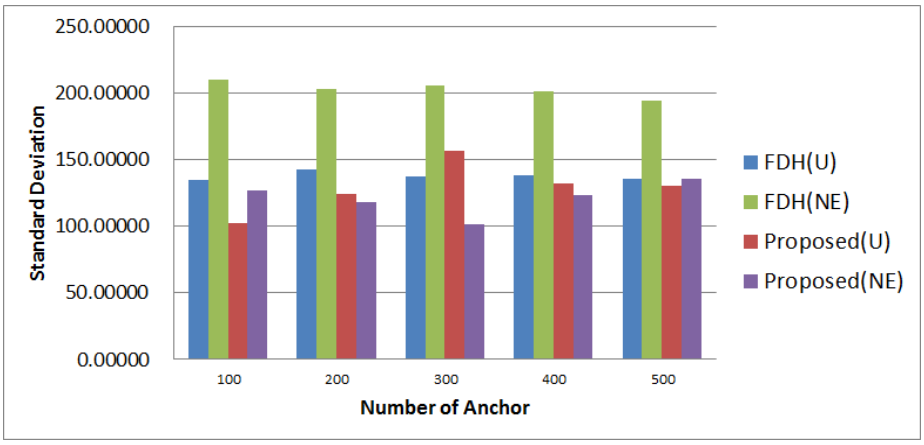


(b) Data Encryption Time with Skewed and Gaussian Data

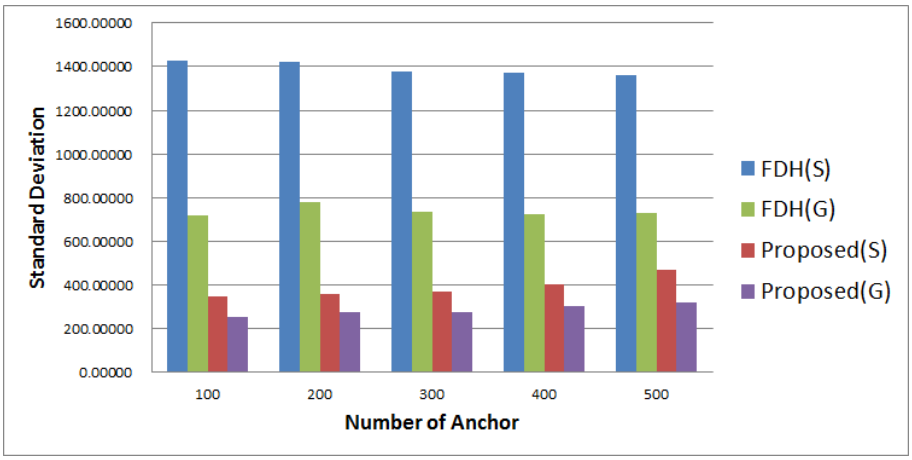
Figure 9. Data Encryption Time with Data Distribution

4.2. Degree of Data Distribution

To verify the effectiveness of our anchor selection algorithm, we measure the number of POIs in clusters with various data distributions. Our scheme shows 20~30% lower standard deviation than that of the existing scheme, as shown in Figure 10. This is because our scheme selects the anchors by considering the data distribution. Our scheme also re-constructs the anchors by using our split and merge algorithm. Whereas, the existing FDH scheme selects anchors randomly without considering the data distribution.



(a) Standard Deviation with Uniform and Real Data



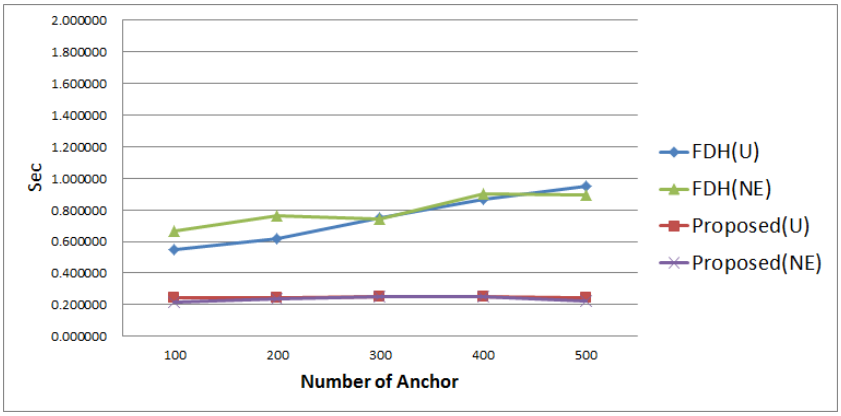
(b) Standard Deviation with Uniform and Real Data

Figure 10. Standard Deviation POI in Anchor Bound with Various Data Distribution

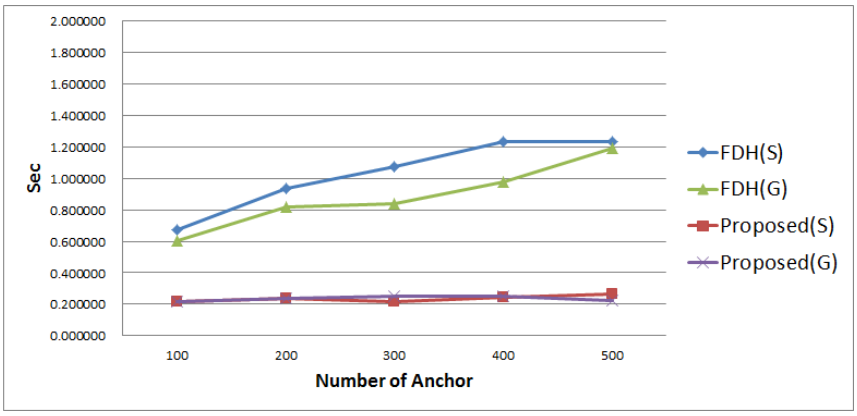
4.3. k Nearest Neighbor Query Processing Time

Figure 11 shows the *k* nearest neighbor(*k*NN) query processing time with varying number of anchors. When the number of anchors is 300 and k is 5, the query processing time of our scheme is 0.21 seconds whereas the existing FDH scheme requires 0.8 seconds. From the result, it is shown that our scheme outperforms the existing work up to 4 times. This is because our scheme reduces the computation cost by using both bit operations and the hash-based signature index.

Table 2 shows the kNN query processing time in uniform data varying with the different value of k. In case that k are 10 and 20, the query processing time of our algorithm is 0.2983 and 0.3472 seconds whereas HSD* requires 0.8648 seconds and 1.0257 seconds, respectively. From above results, our algorithm shows the better performance on k-NN query processing time as FDH.



(a) Query Processing Time with Uniform and Real Data



(b) Query Processing Time with Skewed and Gaussian Data

Figure 11. Query Processing Time with Data Distribution

Table 2. kNN Query Processing Time

k	FDH	Proposed
5	0.72317	0.2521
10	0.86447	0.2983
15	0.90148	0.3229
20	1.02571	0.3472

5. Conclusion

With the development of cloud computing, the interest on spatial database outsourcing has been sharply increasing. Therefore, researches for protecting location data privacy in outsourced databases have been actively performed. However, the existing schemes are weak to access original data because they do not consider data distribution. In this paper, we propose a data encryption scheme that groups data with anchors and transforms them into bitmap information. To enhance the query processing performance, we introduce an algebraic coding based hash index to retrieve bitmap at a time. Through performance evaluation, it is shown that proposed method outperforms the existing method in terms of range query processing time up to 15 times while providing similar performance in returning number of false positive data.

As a future work, we will extend the proposed algorithm to support variety of query types (*e.g.*, range skyline queries).

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2014065816).

References

[1] X. Jiang, J. Gao, T. Wang, and D. Yang, “Multiple sensitive association protection in the outsourced database”, Database Systems for Advanced Applications, 2010.
[2] C. Valentina, et al. "Selective data outsourcing for enforcing privacy." Journal of Computer Security 19.3 2011.
[3] Sacharidis, K. Mouratidis and D. Papadias, “k-Anonymity in the Presence of External Databases”, IEEE Transactions on Knowledge and Data Engineering, 2010.
[4] Zhou, M., Mu, Y., Susilo, W., Yan, J., & Dong, L. (2012). Privacy enhanced data outsourcing in the cloud. Journal of Network and Computer Applications, 35(4), 1367-1373.
[5] S. Haber, W. G. Horne, T. Sander and D. F. Yao, “Privacy-Aware Verification of Aggregate Queries on Outsourced Databases with Applications to Historic Data Integrity”, Privacy Enhancing Technologies 2009.
[6] Y. Yang, D. Papadias, S. Papadopoulos and P. Kalnis, “Authentcated Join Processing in Outsourced Databases”, ACM SIGMOD, 2009.
[7] A. Singh, M. Srivatsa and L. Liu, “Search-as-a-Service: Outsourced Search over Outsourced Storage”, ACM Transactions on the Web, 2009.
[8] A. Singh, L. Liu and S. Jose, “SHAROES: A Data Sharing Platform for Outsourced Enterprise Storage Environments”, IEEE International Conference on Data Engineering, 2008.
[9] L. Xiong, S. Chitti and L. Liu, “Protecting Data Privacy in Outsourcing Data Aggregation Services”, ACM Transactions on Internet Technology, 2007.
[10] Anil, Swapna Lia, and Roshni Thanka. "A survey on security of data outsourcing in cloud." International Journal of Scientific and Research Publications (IJSRP), 2013.
[11] Y. Yang, S. Papadopoulos, D. Papadias and G. Kollios, “Spatial Outsourcing for Location-based Services”, IEEE International Conference on Data Engineering, 2009.
[12] Emekci, Fatih, et al. "Dividing secrets to secure data outsourcing." Information Sciences 2013.
[13] M. L. Yiu, G. Ghinita, C. S. Jensen and P. Kalnis, “Outsourcing of Private Spatial Data for Search Services”, IEEE International Conference on Data Engineering, 2009.
[14] M. L. Yiu, G. Ghinita, C. S. Jensen and P. Kalnis, “Enabling Search Services on Outsourced Private Spatial Data”, VLDB Journal, 2010.
[15] A. Khoshgozaran and Cyrus Shahabi, "Private Buddy Search: Enabling Private Spatial Queries in Social Networks", IEEE International Conference on Computational Science and Engineering, 2009.
[16] Man Lung Yiu, Ira Assent Christian S. Jensen, “Outsourced Simiarity Search on Metric Data Assets,” IEEE Transactions on Knowledge and Data Engineering, 2010.
[17] Gutscher, A, "Coordinate transformation - a solution for the privacy problem of location based services?", Parallel and Distributed Processing Symposium, 2006. IPDPS 2006.
[18] Hamming, Richard W. "Error detecting and error correcting codes," Bell System Technical Journal 29 (2): 147–160, 1950.
[19] P. Ciaccia, M. Patella, and P. Zezula, "M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces," Proc. Very Large Databases(VLDB), pp. 426-435,1997.

Authors

Min Yoon, he is a Ph.D candidate in the Chonbuk National University. He received the B.S and M.S degrees in Chonbuk National University in 2009 and 2011, respectively. His research interests include security and privacy of sensor network and database outsourcing.

Miyoung Jang, she is a Ph.D candidate in the Chonbuk National University. She received the B.S and M.S degrees in Chonbuk National University in 2010 and 2011, respectively. Her research interests include security and privacy of MapReudce framework and cloud computing.

Young-Sung Shin, he is a Ph.D candidate in the Chonbuk National University. He received the B.S and M.S degrees in Chonbuk National University in 2011 and 2013, respectively. His research interests include big data analysis and distributed parallel processing.

Jae-woo Chang, he is a professor in the Department of Information and Technology, Chonbuk National University, Korea from 1991. He received the B.S. degrees in Computer Engineering from Seoul National University in 1984. He received the M. S. and Ph. D degrees in Computer Engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1986 and 1991, respectively. During 1996–1997, he stayed in University of Minnesota for visiting scholar. And during 2003–2004, he worked for Penn State University (PSU) as a visiting professor. His research interests include sensor networks, spatial network database, context awareness and storage system.