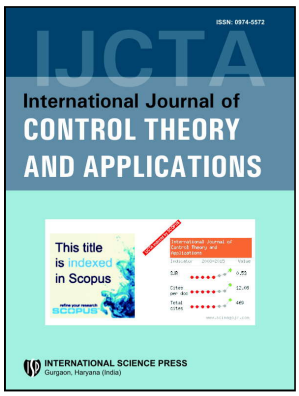

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 14 • 2017

Similarity Search through Secure K-NN and LSH in Cloud

S. Vikram Phaneendra[1], E. Madhusudhana Reddy[2] and S. Ramakrishna[3]

[1] *Research Scholar, Computer Science and Engineering J.N.T.U.K, Kakinada, India, Email: vikramphaneendra@gmail.com*
[2] *Professor, Computer Science & Engineering DRK College of Engineering & Technology, India, Email: e_mreddy@yahoo.com*
[3] *Professor, Computer Science, Sri Venkateswara University, Tirupati, India, Email: drsramakrishna@yahoo.com*

Abstract: Emerging cloud computing technology provides measured service for individual users and organizations, thereby cutting costs. For privacy reasons data is kept or managed in encrypted form only in the cloud, data retrieving is based on a plain text keyword search. Cloud data search services can enable only when data is encrypted but previous work is focused on single or Boolean keyword search which is an overhead. The present work is focused on multi keyword search over an encrypted data without privacy breaks which is a challenging problem. In order to overcome this problem, we can propose a secure K-NN technique for privacy and Locality Sensitive Hashing (LSH) algorithm which is based on secure index for fast similarity search over cipher text data. For safe cloud data utilization system, we establish a group of strict privacy requirements. Eventually, these techniques are effective and efficient in terms of communication and computation overhead.

Keywords: Locality Sensitive Hashing, secure K-NN technique, cloud computing

1. INTRODUCTION

Cloud data search services can enable only when data is encrypted but previous work is focused on single or Boolean keyword search which is an overhead. After this MRSE which is multi-keyword ranked search over encrypted cloud data is proposed by using this privacy breaks are occurring. To meet the test of supporting such multi keyword semantic without security ruptures, they have an essential thought for the MRSE utilizing secure internal item processing, which is adjusted from a protected k-closest neighbor (K-NN) method. After that, give two fundamentally enhanced MRSE conspires in a regulated way to accomplish differentiation stringent protection necessities in two danger models with expanded assault capacities. On a different front, the research on top-k retrieval in database community is also loosely connected to our problem. In order to overcome this problem, a secure K-NN technique is proposed for providing privacy and Locality Sensitive Hashing (LSH) algorithm which is based on secure index for fast similarity search over cipher text data.

2. RELATED WORK

There are many research conducted in the area of improving the efficiency and quality of cooperative data collection.

2.1. Locality Sensitive Hashing (LSH)

The LSH method and its variants have been applied to computational problems in some areas like web clustering or computer vision. The principal goal of the algorithm is to index a collection of n points using hash functions that ensure that the collision (i.e. Similarity between objects) is higher to objects that are closer between each other than four points that are far away. Then, the nearest neighbors can be determined by hashing the query point and returning the elements of the bucket which would contain that point [1]. Locality sensitive hashing is a central large scale data processing algorithm that is, designed to function on objects in higher dimensions. Create a family of functions that hash objects into buckets such that objects that are similar will be hashed to the same bucket within a high possibility.

2.2. Hashing

Hashing is one of the popular solutions for approximate nearest neighbor search. In general, hashing is an approach of transforming the data item to a low dimensional representation, or equivalently a short code consisting of a sequence of bits. The application of hashing to approximate nearest neighbor search includes two ways: indexing data items using hash tables that are formed by storing the items with the same code in a hash bucket, and approximating the distance using the one computed with short codes. The hashing approach aims to map the reference and/or query items to the target items so that the approximate nearest neighbor search can be efficiently and accurately performed using the target items and possibly a small subset of the raw reference items. The target items are called hash codes (also known as hash values, simply hashes).

The LSH calculation has been connected effectively to rapidly discover closest neighbors in substantial databases. As opposed to discovering accurate matches as customary hashes would, LSH considers the region of the focuses so that adjacent focuses stay close-by. Cases of such applications incorporate discovering copy pages on the Web, picture recovery, and music recovery.

2.3. K- Nearest Neighbor Search

The Nearest neighbor search is defined as: Given a query item q, the goal is to find an item NN (q), called nearest neighbor, from a set of items $X = \{x_1, x_2, \cdots, x_N\}$ so that $NN(q) = \text{argmin } x \in X \text{ Dist } (q, x)$, where Dist (q, x) is a distance computed between q and x. A straightforward generalization is a K-NN search, where K-nearest neighbors (KNN (q)) are needed to be found [8].

The problem is not fully specified without the distance between an arbitrary pair of items x and q. As a typical example, the search (reference) database X lies in a d dimensional space Rd and the distance is induced by an ls norm, kx – qks = (Pd i=1 $|x_i - q_i|$s) 1/s. The search problem under the Euclidean distance, i.e., the l2 norm, is widely studied [2]. Other notions of search database, such as each item formed by a set, and distance measure, such as $\ell 1$ distance, cosine similarity and so on are also possible.

3. PROBLEM STATEMENT AND PROPOSED APPROACH TO SOLUTION

3.1. Implementation of LSH

Coordinate matching like K-nearest neighbor Technique, which provides many matches as given multi keyword and provides an efficient similarity measure to refine the accurate results. Multi-keyword search with ranked over an encrypted cloud data while preserving privacy in the cloud. The basic idea of Locality Sensitive Hashing algorithm is to use a group of hash functions to map objects into several buckets such that similar objects share a bucket with high probability.

3.2. Description

The LSH technique was introduced by Indyck and Motwani with the purpose of creating algorithms to solve the K-Nearest Neighbor Search. The principal goal of the algorithm is to index a collection of n

points using hash functions that ensure that the collision (i.e. Similarity between objects) is higher to objects that are closer between each other than four points that are far away. Then, the nearest neighbors can be determined by hashing the query point and returning the elements of the bucket which would contain that point.

Then, the nearest neighbors can be determined by hashing the query point and returning the elements of the bucket which would contain that point. Locality sensitive hashing is a basic large -scale data processing algorithm that is designed to function on objects in higher dimensions. Create a family of functions that hash objects into buckets such that objects that are similar will be hashed to the same bucket within a high possibility.

3.3. Hashing

Hashing is one of the popular solutions for approximate nearest neighbor search. In general, hashing is an approach of transforming the data item to a low dimensional representation, or equivalently a short code consisting of a sequence of bits. The application of hashing to approximate nearest neighbor search includes two ways: indexing data items using hash tables that are formed by storing the items with the same code in a hash bucket, and approximating the distance using the one computed with short codes.

The hashing approach aims to map the reference and/or query items to the target items so that approximate nearest neighbor search can be efficiently and accurately performed using the target items and possibly a small subset of the raw reference items. The target items are called hash codes (also known as hash values, simply hashes). In this paper, we may also call it short/compact code interchangeably. Formally, the hash function is defined as: $y = h(x)$, where y is the hash code and h (·) is the function. In the application to approximate nearest neighbor search, usually several hash functions are used together to compute the hash code: $y = h(x)$, where $y = [y_1\ y_2 \cdots y_M]T$ and $[h_1(x)\ h_2(x) \cdots hM(x)]T$. Here we use a vector y to represent the hash code for presentation convenience. There are two basic strategies for using hash codes to perform nearest (near) neighbor search: hash table lookup and Fast distance approximation.

3.4. Hash table lookup

The hash table is a data structure that is composed of buckets, each of which is indexed by a hash code. Each reference item x is placed into a bucket h(x). Different from the conventional hashing algorithm in computer science that avoids collisions (i.e., avoids mapping two items into some same bucket), the hashing approach using a hash table aims to maximize the probability of collision of near items. Given the query q, the items lying in the bucket h(q) are retrieved as near items of q.

To improve the recall, L hash tables are constructed, and the items lying in the L (L' , $L' < L$) hash buckets h1 (q), · · ·, hl (q) are retrieved as near items of q for randomized R-near neighbor search. To guarantee the precision, each of the L hash codes, Yi, needs to be a long code, which means that the total number of the buckets is too large to index directly. Thus, single non-empty buckets are retained by resorting to convectional hashing of the hash codes hl (x).

3.5. Fast distance approximation

The direct way is to perform an exhaustive search: compare the query with each reference item by fast calculating the distance between the query and the hash code of the reference item. Such that retrieve the reference items with the smallest distances as the candidates of nearest neighbors, which is usually followed by a reranking step: rerank the nearest neighbor candidates retrieved with hash codes according to the true distances computed using the original features and attain the K nearest neighbors or R-near neighbor. This strategy exploits two advantages of hash codes. The first one is that the distance using hash codes can be efficiently computed and the cost is much smaller than that of the computation in the input space. The second one is that the size of the hash codes

is much smaller than the input features and hence can be loaded into memory, a resulting the disk I/O cost reduction in the case the original features are too large to be loaded into memory.

One practical way of speeding up the search is to perform a non-exhaustive search: first retrieve a set of candidates using inverted index and then compute the distances of the query with the candidates using the short codes. Other research efforts include organizing the hash codes with a data structure, such as a tree or a graph structure, to avoid exhaustive search.

3.5.1. Processing

1. Choose L functions g_j, j=1,2, . . ., L by setting $g_j = (h_{ij}, h_{2j}, . . ., h_{kj})$, where h_{ij}, h_{2j}, . . ., h_{kj} are chosen at random from LSH family H.
2. Construct hash tables, where for each j=1, 2, …, l the j^{th} hash table contains the dataset points hashed using the function g_j.

3.5.2. Query algorithm for a quering point q

1. for each j=1, 2, . . ., l
   - I. retrieve the points from the bucket gj (q) in the jth hash table.
   - II. for each of the retrieved point, compute the distance from q to it, and report the point if it is a correct answer (near neighbor)
   - III. (optional) Strop as soon as the number of reported points is more than l'

3.6. Nearest Neighbor Search

The Nearest neighbor search is defined as: Given a query item q, the goal is to find an item NN (q), called nearest neighbor, from a set of items X = {x1, x2, · · · , xN} so that NN(q) = arg min x ∈ X Dist (q, x), where Dist (q, x) is a distance computed between q and x [8]. A straightforward generalization is a K-NN search, where K-nearest neighbors (KNN (q)) are needed to be found. The problem is not fully specified without the distance between an arbitrary pair of items x and q.

The problem is not fully specified without the distance between an arbitrary pair of items x and q. As a typical example, the search (reference) database X lies in a d dimensional space Rd and the distance is induced by an ls norm, kx – qks = (Pd i=1 |xi – qi|s) 1/s. The search problem under the Euclidean distance, i.e., the l2 norm, is widely studied. Other notions of search database, such as each item formed by a set, and distance measure, such as ℓ1 distance, cosine similarity and so on are also possible.

4. LOCALITY SENSITIVE HASHING

A LSH hash functions, family H can be used to elaborate an algorithm to solve the approximate search of the nearest-neighbor. The algorithm is formed by two stages: The preprocessing step and the query algorithm. In the pre-processing step the data structure is built from the input dataset. L hash functions are extracted $h!$, j = 1,2,3…L and a different hash table is created per function. The jth table (with j = 1,2,3…L) contains the characteristics of the data set hashed using the function $h!$.

To get the index of the bucket where the feature should go. Within each hash table 5ØWÜ, all features with the same hash value should go to the same bucket.

4.1. LSH Algorithm

There are two parameters for any LSH method: L and T. L is the total number of hash functions used, if there are many functions the runtime is increased, but there are more probabilities of finding similar elements and, therefore,

Table 1
Example of using multiple tables.

			Hash Table's Buckets		
Hash table1		*P1*	*P2 P3*	*P4*	
Hash table2	*P1*		*P2 P3 P4*		
Hash table3	*P1 P3*		*P4*		P2

the bucket size is smaller. The parameter T is the number of hash tables used. A large number of tables increase the performance at the expense of increasing the runtime. Using multiple hash table s allows us to improve the performance in terms of search, i.e., finding all possible nearest neighbors that would have not been found using a single table.

For example, above figure, if only Hash table 1 was used, the nearest neighbor to *P*! Would be only *P*!, whereas if we use all the tables we find that the points *P*!*P*! And *P*!! We also near *P*! in the query step of the algorithm.

The LSH type of algorithm, for each hash function as a single dimension of the data is chosen uniformly at random, and a single threshold value is drawn uniformly over the data range in that dimension.

The e2LSH algorithm, for each hash function, a random line with independent, normally distributed coefficients is generated and the data are projected to the line and shifted. Then, the range is divided into "cells" of a specific width; the hash value is determined by the cell into which a projected and shifted value falls.

We chose to work with the basic implementation of LSH: LSH type. The code needs to keep around the original data, in order to go back from indices to actual points because LSH will only index the data. There are three basic parameters to play with: the bucket's size, the number of tables and the number of bits of the hash functions.

Each characteristic of vector mapped into buckets through composite hash functions. Consider, one bucket identifier of the index and that feature is contained in the data items which are the members of it. A bit vector of size ℓ is the Bucket content is simply, where ℓ is the aggregate number of informative things to be put away on the server. Assume every information thing is doled out an identifier from 1 to ℓ. The following four steps that are included in secure LSH indexes are:

- We will use a family of hash functions such that close points tend to hash to the same bucket.
- Put all points of P in their buckets, ideally we want the query q to find its nearest neighbor in its bucket
- A family H of functions are locality sensitive with respect to a similarity function $0 \leq \text{sim}(p,q) \leq 1$ if
- Pr[h(p) = h(q)] = sim(p,q)
- Them: If there exists a family H of hash functions such that
- Pr[h(p) = h(q)] = sim(p,q)
- Then d(p,q) = 1-sim(p,q) satisfies the triangle inequality

Secure LSH Index: To utilize the drawing in properties of LSH in the association of the mixed data, we propose a secured LSH rundown and a closeness searchable symmetric encryption plot on top of this record. Additionally, we modify the extensively recognized adaptable semantic security significance of Curtmola et. al. In a searchable symmetric encryption scheme and show the security of the proposed arrangement under the balanced definition.

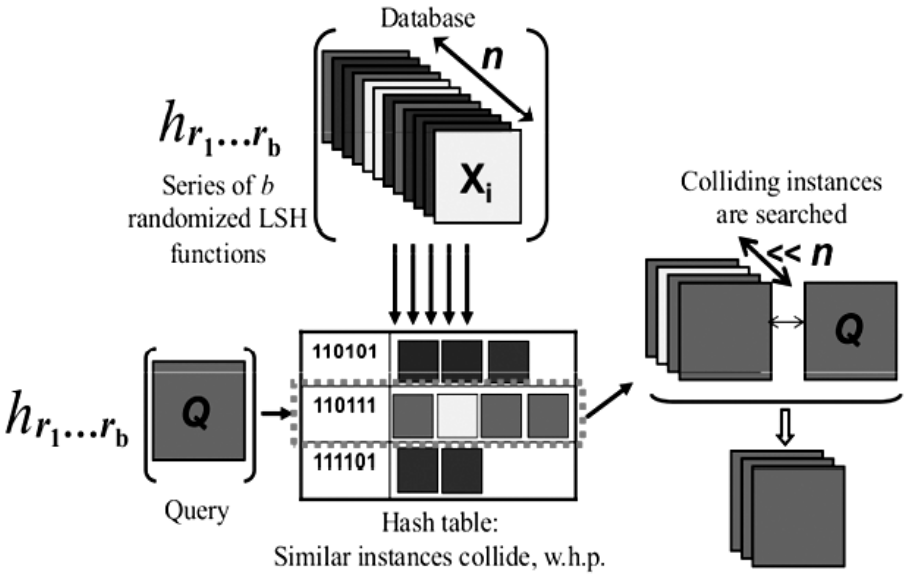

Figure 1: Diagrammatic explanation LSH algorithm.

The LSH calculation has been connected effectively to rapidly discover closest neighbors in extensive databases. As opposed to discovering careful matches as routine hashes would, LSH considers the area of the focuses so that close-by focuses stay adjacent. Samples of such applications incorporate discovering copy pages on the Web, picture recovery, and music recovery.

Advantages of proposed system:

- Communication overhead is low.
- Computation overhead is low.
- It is providing fast similarity search results.
- It provides multiple keywords searched.
- It provides privacy on over encrypted data.

5. METHODOLOGIES IN PROPOSED SYSTEM

Algorithm: **Algorithm for Locality sensitive hashing**

Require: D: data item collection, g: λ composite hash function,

ψ : Security parameter, : E_v Encrypted bit vectors,

$k_{payload}$: Secret key, t: score limit, L_{id}: list of data identifier,

E_d : Encrypted data collection, C: encrypted data item,

k_{coll} **:** Secret key

$$k_{id} \leftarrow keygen(\varphi)$$

$$k_{payload} \leftarrow keygen(\varphi)$$

For all $D_i \in D$ **do**

$F_i \leftarrow exractfeatures\,of\ D_j$

For all $f_{ij} \in F_i$ **do**

$\vec{f}_{ij} \leftarrow apply\,metric\,space\,translation\,on\,f_{ij}$

For all $g_k \in g$ do

If $g_k\left(\vec{f}_{ij}\right) \notin$ bucket identifier list **Then**

Add $g_k\left(\vec{f}_{ij}\right)$ to the bucket identifier list

Initialize $V_{g_k\left(\vec{f}_{ij}\right)}$ as a zero vector of size $|D|$

Increment recordCount

End if

$V_{g_k\left(\vec{f}_{ij}\right)}\left[id\left(D_j\right)\right] \leftarrow 1$

End for

End for

End for

For all $B_k \in$ bucket identification list **do**

$V_{Bk} \leftarrow retrival\ \ payload\ of\ \ B_k$

$\pi_{Bk} \leftarrow Enc_{k_{id}}\left(B_k\right)$

$\sigma V_{Bk} \leftarrow Enc_{k_{payload}}\left(V_{Bk}\right)$

$\left(\pi_{Bk}, \sigma V_{Bk}\right)\ to\ I$

For all $V_{BK} \in E_v$ **do**

$V_{BK} \leftarrow Dec_{k_{payload}}\left(\sigma V_{BK}\right)$

For $i \leftarrow 1 to |V_{BK}|$ **do**

If $V_{BK}[i]=1$ **Then**

Add i to the candidate identifier list increment score (i)

End if

End for

End for

$idList \rightarrow select\ identifiers\ \ from\ candidate\ with\ top\ t\ highest\ score\ send\ idList\ to\ server$

For all $id \in L_{id}$ **do**

If $(id, \phi_{id}) \in E_D$ **Then**

Send ϕ_{id} to client

End if

End for

For all $\phi_{id} \in C$ **do**

$item \leftarrow Dec_{K_{coll}}(\phi_{id})$

End for

End for

6. CONCLUSION

In this paper, we have solved the problem of multi-keyword rank search over encrypted data through locality sensitive hashing and nearest neighbor search. For privacy reasons data is kept or managed in encrypted form only in the cloud, data retrieving is based on a plain text keyword search. Cloud data search services can enable only when data is encrypted but previous work is focused on single or Boolean keyword search which is an overhead. Proposed a secure K-NN technique for privacy the documents are made into chunk in servers and LSH based security index and a search scheme to enable fast similarity search in the context of encrypted data. By using this system are the pertinence of the wanted report is resolved that the technique is built up by the Top K Query. The consequences of that system deliver the obliged K documents to the end client. Finally, it achieves low overhead on communication and computation.

REFERENCES

[1] 2J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "FuzzyKeyword Search Over Encrypted Data in Cloud Computing,"Proc. IEEE INFOCOM, Mar. 2010.

[2] B.Bhaskar, E. Madhusudhana Reddy,"A Novel Load Balancing Model Using RR Algorithm for the Cloud Computing", International Journal of Computer Science and Information Technologies (IJCSIT) (ISSN:0975-9646),Volume 5, Issue 6, 2015, PP 7652-7655.

[3] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data," IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 8, pp. 1467- 1479, Aug. 2012.

[4] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data,"IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 8, pp. 1467-1479, Aug. 2012.

[5] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-PreservingPublic Auditing for Data Storage Security in Cloud Computing,"Proc. IEEE INFOCOM, 2010.

[6] D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT), 2004.

[7] E. Shen, E. Shi, and B. Waters, "Predicate Privacy in EncryptionSystems," Proc. Sixth Theory of Cryptography Conf. Theory of Cryptography (TCC), 2009.

[8] G Reddy Rani, E. Madhusudhana Reddy, "Resemblance Search Through Secure KNN and LSH in Cloud", International Journal of Applied Engineering Research (IJAER) (ISSN: 0973-4562),Volume 10, No. 82, 2015, PP 1-7.

[9] K.Kishore, E. Madhusudhana Reddy, "Outsourcing In Cloud Computing Using Homomorphic Encryption Potentials" International journal of Advanced Scientific and technical Research (ISSN: 2249-9954), August 2012, Issue 2, Volume 4, PP 57-64.

[10] M. Ravi kumar, E. Madhusudhana Reddy, "Auditing Framework Service for Efficient Secure Data Storage in Multi- cloud", International Journal of Computer Science and Information Technologies (IJCSIT) (ISSN:0975-9646),Volume 6, Issue 2, 2015, PP 1181-1183.

[11] N. Cao, S. Yu, Z. Yang, W. Lou, and Y. Hou, "LT Codes-BasedSecure and Reliable Cloud Storage Service," Proc. IEEE INFOCOM,pp. 693-701, 2012.

[12] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Privacy preserving Query over Encrypted Graph-Structured Data in Cloud Computing," Proc. Distributed Computing Systems (ICDCS), pp. 393-402, June, 2011.

[13] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k Retrieval from a Confidential Index," Proc. 12th Int'l Conf. Extending Database Technology (EDBT '09), pp. 439-449, 2009.

[14] Y. Hwang and P. Lee, "Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-User System," Pairing, vol. 4575, pp. 2-22, 2007.