FRORSS: Fast Result Object Retrieval using Similarity Search on Cloud

Raghavendra S*, Nithyashree K*, Geeta C M*, Rajkumar Buyya†, Venugopal K R*, S S Iyengar‡ and L M Patnaik§
* University Visvesvaraya College of Engineering, Bangalore, India. e-mail: raghush86@gmail.com.
†Cloud Computing and Distributed Systems (CLOUDS) Lab, Department of Computing and Information Systems,
The University of Melbourne, Australia.
‡ Florida International University, USA
§Adjunct Professor and INSA Senior Scientist, National Institute of Advanced Studies,
Indian Institute of Science Campus, Bangalore.

***Abstract*—This paper involves a cloud computing environment in which the data owner outsources the similarity search service to a third party service provider. The user provides an example query to the server to retrieve similar data. Privacy of the outsourced data is important because they may be sensitive, valuable or confidential data. The data should be made available to the authorized client/client groups, but not to be revealed to the service provider in which the data is stored. Given this scenario, the paper presents a technique called FRORSS which has build phase, data transformation and search phase. The build phase is about uploading the data; the data transformation phase transforms the data before submitting it to the service provider for similarity queries on the transformed data; search phase involves searching similar object with respect to query object. Experiments have been carried out on real data sets which exhibits that the proposed work is capable of providing privacy and achieving accuracy at a lower value of result measure in comparision with FDH [1].**

***Index Terms*—Cloud computing, Similarity search, Data transformation, Result Measure, FRORSS.**

I. Introduction

THERE is a rapid growth of the volume and diversity of digital data produced by all kinds of commercial, scientific and leisure-time applications;to search for a desired data in such voluminous data set is a tedious task. The complex data types, such as various sensor data, time series data, gene sequence data introduces a natural requirement for search. It is difficult to search such multimedia data using typical keyword search techniques; hence the similarity search [2], [3] comes into picture. With the growing popularity of cloud services, the natural approach is to outsource this task to the cloud environment. Service outsourcing means that the data is provided to third party repositories that are not controlled by the data owner. The outsourced data may be sensitive, confidential, (e.g. medicine data) or otherwise valuable (e.g. collected from a scientific research [4], [5]) and thus the privacy of the data is given at most importance.

The concept of similarity search [2], [6]–[9] is applicable to a wide range of data types together with a practically infinite number of various similarity functions. The time series pattern which has been collected in hourly or weekly basis can be searched by the scientist for similar patterns to indicate an interesting phenomenon. The similarity search can be used for analysis of DNA patterns for understanding gene or gene groups. Similarity search is most prominently used in the field of health care. The healthcare data like X-rays, MRT out- puts, various electric signals are very complex where content-based retrieval [1] using similarity search is helpful. New similarity search applications are constantly being developed, ranging from language translation systems to intellectual property protection. *Result measure (RM)*: Distance between the query object and the result object.

Motivation: Existing solutions offer any one of the following, its either query efficiency at no privacy, or complete data privacy while sacrificing query efficiency. The approach is to shift search functionality to the server, we have the methods Metric Preserving Transformation(MPT) and Flexible Distance-based Hashing(FDH) which do so. The MPT stores relative distance information at the server with respect to a private set of anchor objects which guarantees to fetch exact results, but needs cost of two rounds of communication. The FDH method takes a single round of communication, but does not guarantee to retrieve the exact result. Hence our objective is to retrieve the exact result in just a single round of communication with a reduced result measure.

Contribution: In this paper, we describe a new technique for similarity search on metric data named as Fast Result Object Retrieval using Similarity Search(FRORSS). FRORSS supports for fast retrieval of resultant object with accuracy and it provides privacy for objects by using data transformation steps before uploading to the cloud server. We suggest new technique to overcome the drawbacks of the outsourced similarity search on metric data assets [10]. The implication of the contributions are:

1) FRORSS method is developed to retrive Fast similarity search on metric space data.

978-1-5090-1623-5/16/$31.00 ©2016 IEEE

2) FRORSS algorithm reduces result measure and increases accuracy.
3) The experiment is demonstrated on real time data set(YEAST [4]).

Organisation: The rest of the paper is organized in the following manner; We describe the Related work in Section 2 which gives the pros and cons of similarity search. Background is described in Section 3 which lists some of the existing method for deriving the results from the server. Problem statement and System model describes the working of the system and gives the details about the design goals these are discussed in Section 4. Proposed work describing the implementation of data transformation, build and the search phase is in Section 5.Performance evaluation results are listed in Section 6. Section 7 is discussion about the concept. Conclusions are presented in Section 8.

II. Related Works

We have listed out various work related to similarity search on cloud environment, along with their advantages and diadvantages.

The techniques present in [1], [10]–[12] helps in accomplishing the similarity search over encrypted data and helps in search process to happen while preserving the data privacy. Yiu et al., [10] present methods which transform data, only upon the transformed data similarity queries can be presented to the service provider. FDH (Flexlible Distance Based Hashing) shift the search functionality to the server with a single round of communication, but do not assure of providing accurate result. Kuzu et al., [11] uses LSH for fast similarity search which is tolerant to typographical errors. Zhu et al., [12] also uses the LSH to do similarity search on images. Biolin et al., [1] does a dynamic similarity search using content based retrieval. This ability to search dynamically was helpful in medical industry to retrieve lung images.

Consider [13]–[15] various parameters to do similarity search over encrypted cloud, similarity search based on distances and metric spaces respectively. Xia et al., [13] returns files which are semantically related to the keyword but need to protect semantic information from the files. Hjaltason et al., [14] use M-tree to have a fast similarity search , but suitable only when we have a lage amount data. Amato et al., [15] use inverted files to obtain similarity search . This can be applied to any application which works on any such paradigm and can be modelled using metric space.

Two important concepts in [16], [17] help in carring out similarity search on metric space . Ciaceia et al., [16] proposed M-tree to organize and search large data sets from generic metric space. They perform well in high dimensional space and are more efficient than R* tree . Jang et al., [18] offer a spatial data base encryption scheme that produce a transformed data base by using network distance among POIs (point of intrest). Hence reduces the search range.

III. Background

A. Encrypted Hierarchical Index

It is a hierarchical indexing structure which is built on the Metric Space(MS) object data set; these nodes are encrypted using a symmetric key algorithm and address of the root node is made public. *Mindistance* and *maxdistance* functions are used for the data transformation which makes the algorithm secure. The search service is at the client side. The client request for nodes, decrypts them and applies the search function on these nodes; a new set of nodes are requested again from the server until the required result is found. There exists a considerable traffic between the server and the client due to multiple communication round trips; a reason for increased communication cost. There is an additional overhead on the client due to the search procedure which takes place on its side. The method suffers from relatively very low search efficiency.

B. Metric Preserving Transformation (MPT):

Metric Preserving Transformation makes use of an order-preserving encryption(OPE). The function $f : RR$ is said to be order-preserving iff $x > yf(x) > f(y)$. A set of anchor objects A are selected from the data set P;where each object is assigned an anchor object A_i. The distance is computed for objects with respect to anchor objects. An OPE is applied on these distances and then uploaded to the server. The application of OPE serves as the transformation function. MPT needs two round trip communications during the query phase. The method is sufficiently secure but not well suited for dynamically changing data.

C. Flexible Distance-based Hashing

In the build phase, a set of n anchor objects is chosen and each anchor a_i is assigned a range r_i. Then, for every object o a bitmap of length n is created; the i th bit of this bitmap equals to zero if $d(a_i, o) \leq r_i$, otherwise it equals to 1. The objects are encrypted and are stored on the server along with their bitmap representation. Here bitmap is used as the data transformation function. Here the client is given the liberty to select and vary the θ value; the server returns θ number of objects that are closest to the bit map representation of the query. This approach accomplishes the best communication cost because of the very compact bit map representation; only drawback being it innately supports only approximate search queries.

IV. Problem Statement and System Model

Nearest neighbour search and Range queries are key subclasses of similarity search. Similarity Search is mostly performed on the complex objects like the metric space objects.The previous methods used multiple communication rounds to get the result,hence we work on reducing the communication round to as low as 1.

Objectives:

1) The data owner allows only authorized clients to run search queries on a third party cloud server and get results from it.
2) To avoid the overhead on the client side, the search should be performed on the server side.
3) Data on the server should be stored in a secure way.

A. System Model

The system model has 3 elements as shown in Figure 1: *data owner*, *client* and *server*. Data owner desires that the data be made available to the authorized clients, but to do so he has to host his data on the server which he does not trust. Hence the data owner encrypts the data and uploads to the server. He applies a standard encryption method (symmetric key encryption algorithm e.g., AES) on the data set of original objects; this results in encrypted objects. These encrypted objects along with their *Ids* are uploaded to the server and stored in a relational table (or in the file system). The original objects are subjected to two steps of data transformation DT1 and DT2; the values obtained during the second step of transformation are sent to the server to be indexed. This step is necessary to maintain privacy of the objects uploaded. The original data objects can be anything such as time series, graphs, strings, medical data, and scientific data. Search service is outsourced by the data owner to the server. Data owner shares a secret key with his clients. The clients having the secret key are authorized to use the search service. The client issues a query and must have the key as a proof of its authorization to the server. The server processes the query and returns the similarity result to the client.

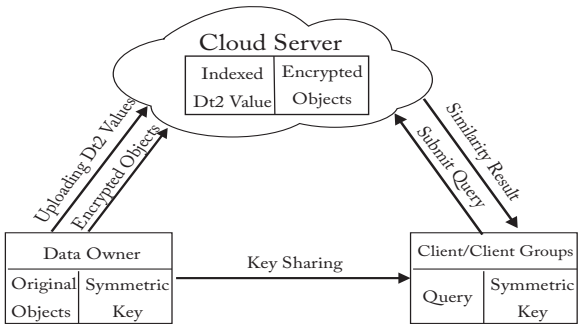

Figure 1. System Architecture

V. PROPOSED WORK

There are three phases in FRORSS technique:

A. Data transformation:

The original objects are transformed and then uploaded to the server. Transformation is important in the aspect of security; objects which are transformed and indexed on the server are having less chances of being understood by the third party like the server or the attackers. The prime concern of the data owner is the privacy of the data kept on these servers and also being able to cater to his authorized clients which lead to the need of data to be transformed. It consists of two stages for Data Transformation(DT):

DT1: The distance function *dist* (a_x, b_y) is said to be a metric if it satisfy symmetry, nonnegativity, and the triangle inequality. This value obtained from the function *dist* (a_x, b_y) is used to compute the dissimilarity between objects a_x and b_y.

The proposed algorithm uses the Euclidean distance function. It is a straight line distance between two points in space. This distance in space turn out to be a Metric Space.

Consider an Euclidean plane; X(x_1 , x_2) and Y(y_1 ,y_2) then the distance between X and Y or the distance between Y and X is given by:

Table I
NOTATIONS

Symbols	*Definition*
P	Data set (set of original objects.
P'	Set of transformed objects.
p_i	Any object in the data set.
p_i|	Transformed object.
$p.id$	*id* of the object p.
CK	Encryption key.
$ECR(x, CK)$	Encrypting x using CK.
$DCR(y, CK)$	Decrypting y using CK.
$dist_e(a, b)$	Euclidean distance.
$dist_o(a, b)$	Object distance.
θ	Integer value used in query phase.
do_i	i^{th} distance object,$i = 1$ to n.
$dist_i$	i^{th} distance from the anchor object, where $i =$ 1 to n.
Eo_i	i^{th} encrypted object, $i = 1$ to n.
$qdist$	distance of the query object from the anchor object.

Algorithm 1: FRORSS algorithm

Build Phase

input : set of original objects

output : distance for each object in the data set with respect to anchor object

Function: Build (P, CK)

1) Choose an object randomly from the data set P as an anchor object;
2) For each object p_i upto p_n where $p_i \in P$and $p_i! = Anchor$ object A_o do;
3) Compute the $dist_e(A_o, p_i)$;
   Compute the $dist_o$
   `/* distance between anchor object and object; */`
4) Compute $ECR(p_i, CK)$;
   `/* encrypt each object with secret key; */`
5) Send the *tuple<p.id, ECR(p, CK),* $dist >$ to the server.;

$$dist_e(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \quad (1)$$

Example: Let $X(2,3)$ and $Y(4,5)$

$$dist_e(X,Y) = \sqrt{((2-4)^2 + (3-5)^2} = 2.8284 \quad (2)$$

In general, for an *n*-dimensional space, where $X(x_1, \ldots, x_n)$ and $Y(y_1, \ldots, y_n)$ the distance is:

$$dist_e(X,Y) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2} \quad (3)$$

DT2: This stage is required to give a sufficient amount of obfuscation about the values indexed in the server.

$$dist_o = \sqrt{dist_e/S} \quad (4)$$

substituting (2) in (4)

$$dist_o = \sqrt{(2.8284/2)} = 1.189 \quad (5)$$

Assume S to be the size of the data set(in terms of number of dimensions).
As a result the original objects $P(p_1, p_2, \ldots, p_n)$ have been transformed to $P'(p_1', p_2', \ldots, p_n')$

B. Build Phase:

This phase takes as the input data set *P*. The data set *P* can contain *n* number of objects, which can be represented as $P(p_1, p_2, \ldots, p_n)$; these objects may also be called as original objects. This phase takes as the input data set *P* and the encryption key CK. A symmetric key algorithm is used for encryption. The choice of our algorithm is AES, just because of its sheer advantages. The key length is around 128 bits or could have a bigger key length according to the confidentiality and the size of the data set chosen.AES is easy to implement, more secure and less prone to attacks.

The build phase mainly occurs at the data owner side. Any random object $p_i \in P$ is selected to be the anchor object. The anchor object is an object with reference to which the similarity or the dissimilarity of the query object is found with respect to the other objects in the data set. Each object is given an *id* , *p.id*. Distance computation (done as in stage 1) of all the objects in the data set is computed with respect to the anchor object. These transformed distances are indexed at the server. In the tree structure as shown in Fig 2, every object except the anchor object is linked. The first node/ root node is having the least DT2 value (that is the object which is most nearest to the anchor object), the second one having the second least DT2 value with respect to the anchor object and so on. The data set gets sorted after it gets uploaded to the server. The tree structure might change dynamically during an update because an anchor object is randomly chosen every time during the data set upload. The objects are encrypted and stored at the server side in file system or a database. The data owner sends the *id*, encrypted object and the distance as the tuple.

C. Search phase:

The search phase takes place on the server. The main concern is to search for a most similar object with respect to the query object and at the same time maintaining the privacy of the data stored at the server. The server finds the nearest object with respect to the queried object at the Server side when the query is received from the client:
qdist distance object tree = dist1 with respect to θ// qdist is the query object distance with respect to the anchor object.
qdist distance object tree = dist2 with respect to θ
if θ=1; we consider a value which is greater (qdist distance object tree = dist1) and a value which is lesser (qdist distance object tree = dist2) than the query distance value. These values are fetched from the distance object tree. The value nearest to the qdist is chosen as the resultant object. if θ= 2; we consider two values which are greater (qdist distance object tree = dist1) and two value which is lesser (qdist distance object tree = dist2)than the query distance value. These values are fetched from the distance object tree. Among these four values, one value which is closest to the qdist is chosen as the resultant object.

Note: our aim is to always have θ=1, hence we consider any θ value greater than 1 as a ruled out option.

- If dist1 and dist2 are equal distance from the anchor object then, dist1 from distance object tree is sent to the client.
- If dist1 is closer to the query object than dist2, then dist1 is sent from distance object tree to the client.
- If dist2 is closer to the query object than dist1 then, dist2 is sent from distance object tree to the client.

The below example shows how the algorithm works for various theta values:

Example:
When θ = 1

- If the query object distance with respect to the anchor is 7 and the following are distances in the distance object tree; 3,4,5,8,10.
- 7 is compared with values stored in the tree
- *qdist* $\leq$ distance object tree = *dist1*
  7$\leq$ 8 , hence the dist1=8
- *qdist* $\geq$ distance object tree = *dist2*
  7$\geq$ 5 , hence the *dist2* = 5
- *dist1* is chosen to be sent to the client as it is only one unit far from the query , where as *dist2* is 2 units away from the query
- More closer the distance from the query, more accurate is the result.

When θ > 1; assuming θ=2, two value with respect to *dist1* and *dist2* are chosen from the tree, the nearest among the 4 values is chosen to return to the client. This is a ruled out option since exact result is got when θ=1, θ >1 would increase the overhead.

When θ <1; it is not possible since the result has to be returned to the client.

VI. PERFORMANCE

We evaluate the performance of the FRORSS based on the real world data set: YEAST [4]. The experiment involves a server and a client. The implementation is done using Java on windows platform using *Intell core i3* CPU 3217 U, with a processor speed of 1.80 Hz. The distance tree is hosted on Open shift cloud; Redhat. YEAST is a gene expression data matrix gained from a Microarray experiment on yeast. Each entry signifies the expression level of a specific gene at a specific condition. The data set used is highly enriched for genes of similar function.Yeast expression matrix datasets is taken from [4]. The matrix consists of 8224 rows and 17 columns, each element occupies 4 bytes of data, where -1 in the matrix indicates a missing value.

In FDH algorithm the value of θ should be varied in order to get a result nearer to the query object. Table II shows the result measure of FDH in comparison with the new algorithm. At FDH, lesser the θ value, greater is the result measure, a very high θ value, is needed to make the result measure 0. The FRORSS algorithm returns the exact result at one communication round without the need to vary the value of θ. The result measure remains 0 for any value of θ. The time complexity applicable for the result measure with respect to FDH [10] is O(n) where θ value has to be varied n times to get result measure as zero, in comparision to FRORSS whose complexity is O(1) because result measure is zero at θ = 1.

In Figure 2, FDH algorithm for θ = 10, the graph starts from

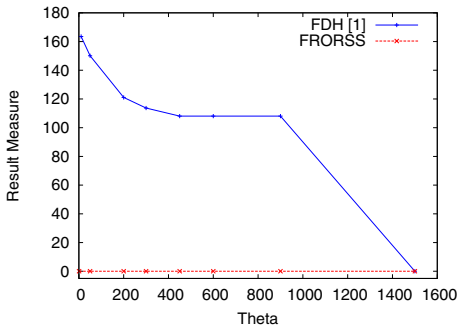

Figure 2. Distance between the Query Object and the Result Object in Comparison with FDH and FRORSS

higher value; as the θ value increases, the result measure reduces slowly. For θ = 600 and θ = 900 their is constant line and hence we increase the θ to a higher value; for example θ = 1500 we see a steep reduction in the result measure in the graph when result measure is zero. The FRORSS algorithm shows no variations and hence the graph remains constant at zero, because the result measure is zero starting from the first attempt.

VII. DISCUSSION

Our aim is to reduce the communication round to as low as 1. We consider communication round as a number of

Table II
COMPARISON OF RESULT MEASURE

FDH [10]		FRORSS	
θ	Result measure	θ	Result measure
10	163.521	1	0
50	150.087	50	0
200	121.07	200	0
300	113.688	300	0
450	108.056	450	0
600	108.056	600	0
900	108.056	900	0
1500	0	1500	0

times the server and the client need to communicate to get to the result object. The distance function is closely related to similarity function. A similarity function is defined over pairs of points which measure the similarity of the two points. The similarity function is inversely related to distance function. If a pair of points is very similar to one another, the distance between them is small. We make use of a distance function (Euclidean distance). DT1(Data Transformation1) and DT2(Data Transformation2) are combinedly used in both build phase and quey phase. We also consider them as steps for data transformation.

- DT1 is used for finding the similarity distance between the objects and also we consider it as the first step to data transformation. The similarity value gained from DT1 is used in the second step of transformation.
- If we store the DT1 values on the server there are possibilities that an intruder would observe the values and the privacy of the data owner would be invaded. Hence we introuduce a second step "DT2" to further transform the DT1 values. Data owner computes DT1 and DT2 on the metric space objects(original objects). To add security, only DT2 values are uploaded to the server by the data owner. The distance object tree stores the DT2 values on the server. Similarly the client computes the DT1 and DT2 values , but sends only the DT2 values to match for a similar object on the server.

Theta (θ) is a measure used to increase the accuracy of the results(result object) fetched from the server.

- Our aim is to have θ=1, hence we consider any θ value greater than 1 as a ruled out option.
- θ value is mentioned by the client when it communicates to the server in order to have better accuracy. Client considers starting from θ =1 and it does not exceed 1.

When θ=1

- Only one tuple(result object) which is exactly similar to the query object is fetched from the server in a single communication round.
- RM=0 the object which is exactly similar to the query object is fetched from the server.

VIII. Conclusion

In this paper, we propose a similarity search technique which works on cloud computing environment without letting the server invade the privacy of the data stored in it. Existing solutions offer tradeoffs between privacy, query accuracy and communication cost. We introduce a concept which outsources the search service to the server. The proposed FRORSS algorithm performs data transformation on the original objects before storing it on the server which ensures privacy. It also retrieves accurate result at a θ value as low as 1. We express its efficiency by experimenting on real data set. It can be applied to any application where the similarity search hypothesis can be modelled using metric spaces.

References

[1] M Blessa Binolin Pepsi and K Mala. Similarity Search on Metric Data of Outsourced Lung Images. *in 2013 IEEE International Conference on Green High Performance Computing (ICGHPC)*, pages 1–6, 2013.

[2] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. Similarity Search: The Metric Space Approach. *in Proceedings of the Advances in Database Systems*, 32, 2006.

[3] Zhihua Xia, Yi Zhu, Xingming Sun, and Jin Wang. A Similarity Search Scheme over Encrypted Cloud Images based on Secure Transformation. *International Journal of Future Generation Communication and Networking*, 6(6):71–80, 2013.

[4] Yizong Cheng and George M Church. Biclustering of Expression Data. *http://arep.med.harvard.edu/biclustering/, Ismb*, 8:93–103, 2000.

[5] Jeremy Hubble, Janos Demeter, Heng Jin, Maria Mao, Michael Nitzberg, TBK Reddy, Farrell Wymore, Zachariah K Zachariah, Gavin Sherlock, and Catherine A Ball. Implementation of Genepattern within the Stanford Microarray Database. *Nucleic Acids Research*, 37(suppl 1):D898–D901, 2009.

[6] S Raghavendra, C M Geeta, K Shaila, Rajkumar Buyya, K R Venugopal, S S Iyengar, and L M Patnaik. MSSS: Most Significant Single-keyword Search over Encrypted Cloud Data. *in Proceedings of the 6th Annual Intrernational Conference on ICT: BigData, Cloud and Security*, 2015.

[7] S. Raghavendra, C. M. Geeta, Rajkumar Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik. DRSIG: Domain and Range Specific Index Generation for Encrypted Cloud Data". In *Proceedings of the International Conference on Computational Techniques in Information and Communication Technologies, ICCTICT 2016*. IEEE, March 17-20 2016.

[8] S. Raghavendra, C M Geeta, Rajkumar Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik. MSIGT: Most Significant Index Generation Technique for Cloud Environment. In *Proceedings of the 12th IEEE India International Conference on E^3-C^3(INDICON 2015)*. IEEE, December 11-13 2015.

[9] S. Raghavendra, S Girish, C. M. Geeta, Rajkumar Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik. IGSK: Index Generation on Split Keyword for Search Over Cloud Data. In *Proceedings of the 2015 International Conference on Computing and Network Communications (CoCoNet'15)*, pages 380–386. IEEE, December 16-19 2015.

[10] Man Lung Yiu, Ira Assent, Christian S Jensen, and Panos Kalnis. Outsourced Similarity Search on Metric Data Assets. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):338–352, 2012.

[11] Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu. Efficient Similarity Search over Encrypted Data. *in IEEE 28th International Conference on Data Engineering (ICDE)*, pages 1156–1167, 2012.

[12] Yi Zhu, Xingming Sun, Zhihua Xia, Li Chen, Tao Li, and Daxing Zhang. Enabling Similarity Search over Encrypted Images in Cloud. *Information Technology Journal*, 13(5):824–831, 2014.

[13] Zhihua Xia, Yanling Zhu, Xingming Sun, and Lihong Chen. Secure Semantic Expansion Based Search over Encrypted Cloud Data Supporting Similarity Ranking. *Journal of Cloud Computing*, 3(1):1–11, 2014.

[14] Gisli R Hjaltason and Hanan Samet. Index-Driven Similarity Search in Metric Spaces (Survey Article). *ACM Transactions on Database Systems (TODS)*, 28(4):517–580, 2003.

[15] Giuseppe Amato and Pasquale Savino. Approximate Similarity Search in Metric Spaces using Inverted Files. *in Proceedings of the 3rd international conference on Scalable information systems*, (28):1–10, 2008.

[16] Paolo Ciaccia, Marco Patella, and Pavel Zezula. DEIS-CSITE-CNR. *in Proceedings of the International Conference on Very Large Data Bases*, 23:426–435, 1997.

[17] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order Preserving Encryption for Numeric Data. *in Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 563–574, 2004.

[18] Miyoung Jang, Min Yoon, and Jae-Woo Chang. A k-Nearest Neighbor Search Algorithm for Privacy Preservation in Outsourced Spatial Databases. *International Journal of Smart Home*, 7(3):239–247, 2013.