Privacy-Preserving Search in Data Clouds Using Normalized Homomorphic Encryption

Mohanad Dawoud[1] and D. Turgay Altilar[2]

Department of Computer Engineering, Istanbul Technical University, Istanbul, Turkey
{dawoud,altilar}@itu.edu.tr

Abstract. By the rapid growth of computer systems, many IT applications that rely on cloud computing have appeared; one of these systems is the data retrieval systems, which need to satisfy various requirements such as the privacy of the data in the cloud. There are many proposed Privacy-Preserving search (PPS) techniques that uses homomorphic encryption to process the data after encryption, but these techniques did not take into account the possibility of repetition of some values of the features table (especially zero), even after the encryption, which makes them vulnerable to frequency attacks. On the other hand, the non-inclusion of these values may lead to the ability to infer some statistical information about the data. In this paper, we took the advantages of homomorphic encryption to encrypt the data as well as preventing any ability to infer any kind of information about the data by normalizing the histogram of the features table while maintaining the quality of the retrieval. The results showed that the proposed technique gave better retrieval efficiency than the previously proposed techniques while preventing frequency attacks.

Keywords: data clouds, security, homomorphic encryption, normalization, frequency attacks, data retrieval.

1 Introduction

Recently, and with the quick production of the enterprise systems and the need for competition with highly supported and resource-allocated systems, clouds became essential in the IT industry. Cloud was defined by Buyya in [2] as a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers. By this way, any new or small system can has the same capabilities of the resources (computing, storage, etc) as the enterprise systems in a cheap and scalable way, also the enterprise systems can benefit from the clouds by increasing capacity or adding capabilities, by pay-per-use service, according to their current needs. Nowadays, there are many platforms for the cloud computing that are opened for the users, such as Amazon's EC2, IBM's Smart Business cloud offerings, Microsoft's Azure and Google's AppEngine. Rad et al. surveyed many platforms by comparing their arrangements, foundation and infrastructure services and their main capabilities used in some leading software companies [10]. To realize the cloud, many requirements should be satisfied as shown by Dikaiakos in [4], suitable software/hardware architecture, data management, cloud interoperability, security, privacy, service provisioning and cloud economics are the main requirements; these requirements can be extended into many more specific requirements. Despite the advantages of using clouds and the ability to reduce costs and to improve the productivity, security issues should be handled carefully; they may inhibit wide adoption of the cloud model [1]. Jansen and Grance provided an overview of the security and privacy challenges pertinent to public cloud computing, they pointed out considerations that organizations should consider when they outsource their data, applications and infrastructure to a public cloud environment [7]. According to Zhang et al. [15], the security and integrity of the cloud images are the foundation of the overall security of the cloud. One of the new security related research problems is the Privacy-Preserving Search (PPS) over encrypted data. The importance of this problem comes from being the cloud server untrusted or curious. Fig. 1 shows a simple model of data cloud comprising of three actors: Data Owner, Cloud Server (or simply Cloud) and Client. The Owner is the one who has a large set of data to be searched, she encrypt the data and outsource it with the querying services to the Cloud, the Cloud is responsible of storing and processing the data, while the Client will query the data stored in the cloud using the trapdoors that are given by the Owner, therefore, the following requirements need to be satisfied to achieve the Privacy-Preserving search in such a model:

1. Neither the cloud nor the data owner is allowed to know or to be able to deduce anything about the client's queries.
2. The cloud should process the client's queries.

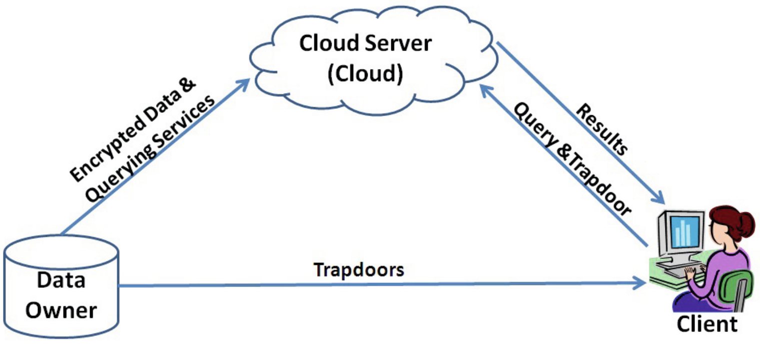

Fig. 1. A simple Model of Data Clouds

Gopal and Singh [5] proposed a technique for Privacy-Preserving search that uses Gentry's Fully Homomorphic encryption [13]. The technique uses the Homomorphic encryption to encrypt the number of occurrences of each keyword in the documents (by the owner) and the query (by clients) using the same key, so, if the cloud does not know the key, it will do calculations using the encrypted data and query without being able to know what they mean. Cao et al. [3] proposed a multi-keyword ranked search technique. The idea depends on encrypting these vectors by some operations that includes adding dummy keywords, splitting of the vectors and multiplication by the key (the key consists of one vector and two matrices). On the other side, the client will also apply the same operations (with some changes) on the query vector using the same key before sending it to the cloud, which in turn processes the encrypted vectors (the query and index) to generate the similarity vector. Li et al. [9] proposed a technique for fuzzy keyword search over encrypted data; In this technique, the data owner builds an index by constructing a fuzzy keyword set then computing trapdoor set with a secret key shared between data owner and authorized users, the data owner sends this index to the cloud. To search in the dataset, the authorized user computes the trapdoor set for the query keyword using the same secret key shared between her and the data owner then sends it to the cloud. Upon receiving the search request, the cloud compares them with the index table and returns all the possible encrypted file identifiers according to fuzzy keyword definition explained in their paper. For such techniques, deterministic encryption is needed to give the right matches.

2 Problem Statement

In data mining, Term Frequency (TF) table is used to get feature vectors for the documents (especially text documents). In this paper, we consider a dataset D consists of N documents where D=(d_1, d_2, ..., d_N), and the set of all the ID's of these documents is ID=(id_1, id_2, ..., id_N), the total number of the unique keywords in the entire documents is M, therefore, the set of all the unique keywords is W=(w_1, w_2, ..., w_M). For a TF table, the rows represents ID while the columns represents W, so, TF=[$x_{n,m}$ | $1 \leqq n \leqq N$ and $1 \leqq m \leqq M$], the value of $x_{n,m}$ represents how many times the m'th keyword is found in the n'th document. If the value of an entity $x_{n,m}$ is zero, this means that the n'th document doesn't has the m'th keyword, also, any equal values in one column means that the corresponding documents has the same keyword with equal number of occurrences. Creating TF table generates a lot of entities with zero value; to show that, stopwords are removed from the documents of the "uw-can-data" dataset [6] using three lists of stopwords, Table 1 shows that the ratio of the non-zero entities to the zero entities in the TF table is 1.41%, which means large number of zeros in the table. For retrieval efficiency, Term Frequency–Inverse Document Frequency (TF-IDF) table [11] is used. Also, for security, these entities need to be encrypted. In most efficient Privacy-Preserving Search techniques, the entities of the comparable parts of the index need to be encrypted individually, therefore,

Table 1. Statistics of the keywords in the "uw-can-data" dataset

The total number of keywords in the documents	The total number of different keywords in the documents	The number of non-zero's in the table	The number of zero's in the table	The ratio of the non-zero's to the zero's in the table (non-zeros/ zero's)
91923	21014	91923	6506473	1.413%

if the encryption has to be deterministic, the values in the TF-IDF table will be mapped to new values in the encrypted TF-IDF table, which means a new table with the same statistics but different values, this make the dataset vulnerable to frequency analysis attacks, whatever the value that appears with largest number of times in the encrypted TF-IDF table, it will be considered to represent the zero's in the TF table. During this paper we will call this "zero's attack".

2.1 Zero's Attack

The matrix multiplication, as in Cao et al. [3] technique, may handle the zero's attack problem since each element in the vector will depend on the other elements in the same vector and the corresponding vector in the key matrix, so, elements with zero or high frequently occurred values will have different values after encryption according to the randomness of the key. But, this is not the case with the techniques similar to Gopal et al. [5] since the entities of the features table is encrypted using the same key. Also, for techniques similar to Li et al. [9], where the encrypted keywords are compared to find the matches, it will need the similar keywords before encryption keep similar after encryption which makes it vulnerable to frequency analysis attacks. Therefore the proposed technique has to be developed to prevent this frequency analysis attacks keeping in mind not to affect the properties of homomorphic encryption and the retrieval efficiency.

2.2 Relations between Documents

Technique to be developed should not allow the cloud to deduce any relation between documents from the encrypted index. Including only the keywords with values greater than zero can also give an idea about which keywords are not found in specific documents, which can be considered as threat as in [9] where the index consists of the unique keywords and the document ID's for only the documents that include each of these keywords.

2.3 Retrieval Efficiency

Data retrieval quality depends on many different factors; one of these factors is the way of choosing feature vectors for the documents. According to [12], binary term vectors give lower efficiency than weighted term vectors. Note that Cao used the binary vectors while Gopal used the weighted vectors in their techniques.

3 Suggested Technique

As mentioned before, Cao et al. [3] can hide zero's and high frequently occurred values. However, because of using the binary vectors (beside the dummy keywords), the retrieval efficiency will be lower than weighted term vector algorithms. Therefore, Gopal technique [13] has to be improved to be able to handle the three issues mentioned in Section 2. With reference to Fig. 1, the suggested model is working as follows:

1. Data owner creates the TF table; the keywords in this table are hashed.
2. The names of the documents and the documents themselves are encrypted separately using symmetric or asymmetric key (K_s).
3. TF-IDF is created from the TF table.
4. TF-IDF table is normalized using the technique which will be explained later in this section.
5. The entities of the normalized TF-IDF table are encrypted individually using homomorphic encryption with the same key (K_h), the encrypted TF-IDF table is the index that will be outsourced to the cloud (encrypted data & querying services).
6. K_h and K_s are sent from the data owner to the client (the trapdoors).
7. The client applies the same operations on the query using K_h before sending it to the cloud.
8. The cloud calculates the similarity between the query and the documents using operations on the encrypted data without revealing them.
9. The similarity vector is sent to the client to decrypt it using K_h and find the best matches to be retrieved.
10. The client sent the names to the cloud and the cloud sends the encrypted file that will be decrypted by the client using the secret key K_s.

Prior to explaining the suggested normalizing technique, the need of including zero's as well as hiding these zero's and highly frequented values have to be discussed. Assume that:

1. $Keyword_1$ found in documents 1, 3 and 8 for 5, 12 and 6 times respectively.
2. $Keyword_2$ found in documents 1, 3 and 9 for 3, 1 and 13 times respectively.
3. $Keyword_3$ found in documents 4, 5 and 10 for 7, 9 and 2 times respectively.

Even the keywords, document names and frequencies are encrypted; one can end up with some deductions such as:

1. Document 1 and document 3 are related (contain two common keywords)
2. Document 8 and document 4 are not related (have no common keywords)
3. Document 1 does not contain $Keyword_3$

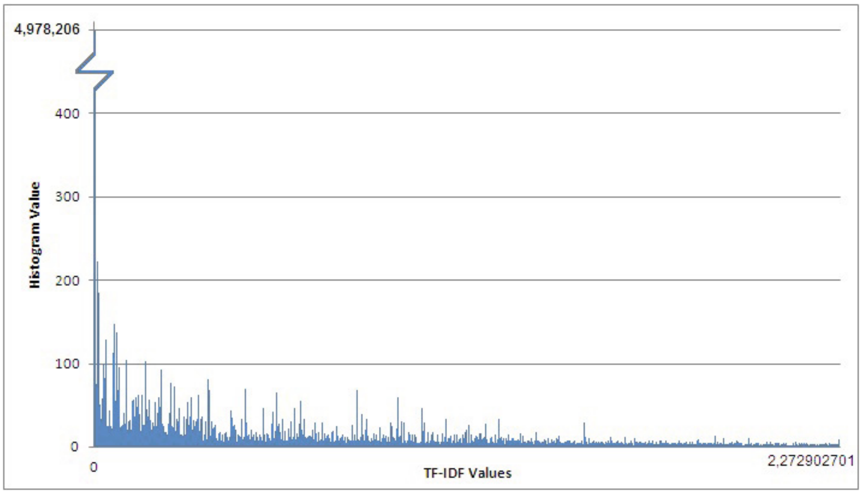

Fig. 2. Histogram for the TF-IDF table of uw-can-data dataset

Even though such a simple example, it is seen that including zeros is necessary to prevent such deductions. Fig. 2 shows the histogram for the TF-IDF table of uw-can-data dataset [6], some TF-IDF values have frequencies more than others, which can be considered as indicators to them in the frequency analysis attacks even after encryption. So, the goal is to normalize these values before encrypting them.

Consider the number of the unique values in the TF-IDF table is Q, then U= (u_1, u_2, ..., u_Q) where U is the set of unique values in the TF-IDF table, in this case, the histogram will be H=(h_1, h_2, ..., h_Q) where h_q represents the number of times that u_q appeared in the TF-IDF table for $1 \leqq q \leqq Q$. To normalize these values, the following steps are done:

1. Order U increasingly in V=(v_1, v_2, ..., v_Q). Values of H will be ordered corresponding to V in HV=(hv_1, hv_2, ..., hv_Q).
2. For each $v_q \in V$, calculate $e_q = (v_{q+1} - v_q)/(hv_q \times k)$, where k is scaling factor that determines the size of difference between the original value and the normalized values (will be discussed later in Section 4). For e_Q, minimum e_q value is taken to be its value.
3. For each $v_q \in V$:

   (a) Define S_q=hv_q-1

   (b) Generate new set v_q' = (v_q'$_0$, v_q'$_1$, ..., v_q'$_{S_q}$) where $0 \leqq s \leqq S_q$ as follows:

   – For $s = 0$ to (S_q)

   • $v_q{}'_s = v_q + s \times e_q$

(c) Replace all the entities in the TF-IDF table that have the v_q value by the elements of the v_q' randomly without repetition.

In this case all the TF-IDF values will be different. Also, even in small difference between the values will be hidden by the encryption process. So, the final step in creating the index for the cloud is to encrypt the entities of the normalized TF-IDF table using Homomorphic encryption [13]; this hides the actual values, but operations on these values are still applicable. To discuss the effect of applying this technique on the retrieval efficiency, the retrieval efficiency of the normalized TF-IDF table is compared with the original TF-IDF table. Average precision value (APV) is used to calculate the retrieval efficiency of the techniques as follows:

1. For each document $d_n \in D$, calculate the precision value pr_n as follows:

$$pr_n = \frac{RetrievedDocuments \ \cap \ RelatedDocuments}{RetrievedDocuments} \tag{1}$$

   Where the number of retrieved documents is equal to the size of the cluster containing the document d_n in the original dataset
2. Calculate the APV as follows:

$$APV = \frac{\sum\limits_{n} pr_n}{NumberofDocuments} \tag{2}$$

4 Simulations and Results

In order to test the suggested technique, we used three different datasets: uw-can-data [6], mini-classicdocs [14] and mini-20newsgroups [8]. Table 2 shows some details about these three datasets. The datasets are prepared before being used by the following steps:

1. html documents are parsed using htmlparser-1.6 to extract the data from them.
2. Stopwords are removed using three different lists of stopwords: Long list, Short list and Google list.
3. Porter stemmer is used to stem the keywords.
4. The datasets are classified using k-means classification with cosine similarity distance.

Using the normalization technique will make all the histogram values of the normalized TF-IDF table equal one. The number of different numbers of the TF-IDF table will be equal to: number of unique keywords $\times$ number of documents To know the effect of normalization on the retrieval efficiency, different values of the factor k are used. As mentioned before, the factor k determines the size of the difference between the original value (v_q) and the expanded set of values (v_q') in the normalization process. The technique was applied on the uw-can-data, mini-20newsgroups and mini-classicdocs datasets separately as follows:

- For z=1 to 10000 increasing by 5:
    - Calculate APV_Z= the APV where k=z.
    - Calculate AV= Average of APV_Z over all z values.

Table 2. Details of the three datasets (uw-can-data, mini-20newsgroups and mini-classicdocs) used in the evaluation of the suggested technique

Dataset	Number of Documents	Number of Classes	Description
uw-can-data	314	10	web pages from various web sites at the University of Waterloo, and some Canadian websites
mini-20newsgroups	400	20	20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups, the number of documents is minimized to 400 documents with the same number of classes
mini-20newsgroups	800	10	Consists of 4 different document collections: CACM, CISI, CRAN, and MED. the number of documents is minimized to 800 documents clustered in 10 classes

Table 3 shows the APV's using the original TF-IDF tables (without normalization) for the three datasets in the first column, which is the case in Gopal et al. [5] technique. The second column represents the APV's for the binary term tables also for the three datasets, which is the case in Cao et al. [3] technique, Finally, the third column represents the average APV's (AV's) for the normalized TF-IDF tables with k = 1 to 10000 increased by 5 for the three datasets, which is the case in the suggested technique in this paper.

5 Analysis

The effectiveness of proposed technique is discussed in this section with regard to the results given in Section 4.

Table 3. Comparison between normalized and unnormalized TF-IDF tables according to AVP and AV values

Dataset	APV without normalization (Gopal Technique)	APV With binary term tables (Cao Technique)	AV value for the normalized TF-IDF tables
uw-can-data	0.175935689	0.150279841	0.183681939
mini-20newsgroups	0.110836309	0.101195467	0,114799958
mini-20newsgroups	0.110236005	0.107274797	0.111710287

Table 4. Comparison between the three discussed techniques

Problem	Gopal Technique	Cao Technique	Suggested Technique
Hiding Zero's	Doesn't hide Zero's	Hides zero's	Hides zero's
Relations Between Documents	Can be deduced	Hard to deduce	Hard to deduce
Retrieval Efficiency	Higher than Cao	Lower than Gopal	Higher than both

5.1 The Effects of the Used Normalization on Privacy

Using normalization gave different values with the same number of appearance in the TF-IDF table which prevents any kind of frequency attacks (discussed in 2.1 and 2.2 subsections). Although the difference between the values may be small before encryption, the Homomorphic encryption will map them to different values.

5.2 The Effects of the Normalization on Retrieval Efficiency

Results show that the retrieval efficiency does not decrease after normalization of the TF-IDF tables. As shown in Table 3, the average of the APV's (AV) after normalization are higher than the precision values before normalization for the three datasets.

5.3 The Effects of this Technique on the Time and Memory Costs

Time cost: The normalization technique will be done once in the setup of the system (which is offline process), all the steps can be done using parallel processors, ordering the histogram increasingly according to the TF-IDF values is O(n log$n$) for n unique keywords. Memory cost: Storing the different values after normalization will be: (number of unique keywords $\times$ number of documents $\times$ size of each unit). Table 4 summarizes the comparison between the two discussed techniques with the proposed technique with regard to the first three problems have been introduced in Section 2.

6 Conclusion

We started with three problems: Zero's attack, hiding relations between documents and conserving retrieval efficiency. We proposed a technique that normalizes the TF-IDF tables; this technique hides the large number of zeros (or any highly frequented values) in the tables as well as any other relation between documents since it keeps the zeros. The technique was applied on three different datasets; results show that the technique improves the retrieval efficiency even with small values. The next step is to find a technique to retrieve only the chosen documents without giving any information about them to both the client and the cloud, or in the case of sending the similarity vector to the client, she will not be able to know anything about the unchosen documents, the technique should also prevent the cloud from guessing any relation between the document lists and the previous queries on the same dataset; this technique should integrate with the suggested technique in this paper to satisfy the needs of a "Privacy-Preserving Search in Data Clouds"

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing (February 2009)
2. Buyya, R.: Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009, p. 1 (May 2009)
3. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Transactions on Parallel and Distributed Systems 25(1), 222–233 (2014)
4. Dikaiakos, M.D., Katsaros, D., Mehra, P., Pallis, G., Vakali, A.: Cloud computing: Distributed internet computing for it and scientific research. IEEE Internet Computing 13(5), 10–13 (2009)
5. Gopal, G.N., Singh, M.P.: Secure similarity based document retrieval system in cloud. In: 2012 International Conference on Data Science Engineering (ICDSE), pp. 154–159 (July 2012)
6. Hammouda, K., Kamel, M.: Web mining data - uw-can-dataset (June 2013), `http://pami.uwaterloo.ca/~hammouda/webdata`
7. Jansen, W., Grance, T.: Sp 800-144. guidelines on security and privacy in public cloud computing. Technical report, Gaithersburg, MD, United States (2011)
8. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the Twelfth International Conference on Machine Learning, pp. 331–339 (1995)
9. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: 2010 Proceedings IEEE INFOCOM, pp. 1–5 (March 2010)
10. Pastaki Rad, M., Sajedi Badashian, A., Meydanipour, G., Ashurzad Delcheh, M., Alipour, M., Afzali, H.: A survey of cloud platforms and their future. In: Gervasi, O., Taniar, D., Murgante, B., Laganà, A., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2009, Part I. LNCS, vol. 5592, pp. 788–796. Springer, Heidelberg (2009)

11. Rajaraman, A., Ullman, J.D.: Data Mining: Mining of Massive Datasets. Cambridge University Press (November 2011) Number 978-1107015357
12. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. In: Information Processing and Management, pp. 513–523 (1988)
13. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
14. Volkan, T.: Data mining research - classic3 and classic4 datasets (January 2012), `http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets`
15. Wei, J., Zhang, X., Ammons, G., Bala, V., Ning, P.: Managing security of virtual machine images in a cloud environment. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW 2009, pp. 91–96. ACM, New York (2009)