Privacy-preserving Data Retrieval using Anonymous Query Authentication in Data Cloud Services

Mohanad Dawoud and D. Turgay Altilar
Computer Engineering Department, Istanbul Technical University, Istanbul, Turkey

Keywords: Privacy, Data Retrieval, Anonymous Authentication, Cloud Computing, Homomorphic Encryption.

Abstract: Recently, cloud computing became an essential part of most IT strategies. However, security and privacy issues are still the two main concerns that limit the widespread use of cloud services since the data is stored in unknown locations and retrieval of data (or part of it) may involve disclosure of sensitive data to unauthorized parties. Many techniques have been proposed to handle this problem, which is known as Privacy-Preserving Data Retrieval (PPDR). These techniques attempt to minimize the sensitive data that needs to be revealed. However, revealing any data to an unauthorized party breaks the security and privacy concepts and also may decrease the efficiency of the data retrieval. In this paper, different requirements are defined to satisfy a high level of security and privacy in a PPDR system. Moreover, a technique that uses anonymous query authentication and multi-server settings is proposed. The technique provides an efficient ranking-based data retrieval by using weighted Term Frequency-Inverse Document Frequency (TF-IDF) vectors. It also satisfies all of the defined security requirements that were completely unsatisfied by the techniques reported in the literature.

1 INTRODUCTION

Information Technology (IT) systems are used steadily in most technology fields. Therefore, the size of data that need to be stored, processed, and transferred through different public, private, or hybrid network systems is increasing rapidly. Recently, cloud computing services have been constituting the best solution to deal with this explosion in the data size. There are many cloud systems that offer different services with high potentials, however, security and privacy still being the main concerns in such systems. The nature of the cloud requires the transfer of the data to unknown locations to store or process them. Storing the data in the cloud systems can be secured by the traditional symmetric or asymmetric encryption algorithms. However, any data mining or retrieval processes need the data, or part of it, to be revealed to the cloud system, which may contradict with conventional security and privacy concepts. Accordingly, many techniques have been proposed to enable the cloud to apply searching processes on the data without revealing them, or, revealing as little as security and privacy rules allow. Figure 1 shows the basic model of such a system. The system consists mainly of three parts: data owner, cloud server (cloud), and client (user). Data owner has a large number of documents that need to be indexed, searched, and partially retrieved by the user, but he does not have the processing and storage capabilities to do that. Cloud has the processing and storage capabilities needed to serve the system, but it is assumed to be “honest-but-curious”. “Honest-but-curious”means that the cloud follows the designated protocol honestly, but curious to infer useful information by analysing the data flow during running the protocol. User needs to retrieve documents related to a queried document (or keywords). Data owner creates indexes for the documents and store the indexes as well as the documents in the cloud in an encrypted form. He also creates trapdoors and send them to the user. User utilizes these trapdoors with the queried document to create a query. He sends the query to the cloud which in turn replies by the related documents.

Suppose that $features(\gamma)$ and $index(\gamma)$ are the features and index of a document γ, respectively. Also, $query(\theta,\tau)$ is the query generated from the document θ and the trapdoor τ. To keep a high level of security and privacy of the data as well as the queries, following requirements need to be satisfied by any proposed protocol:

1. **No Index Pattern:** For any two documents α and β where $features(\alpha) = features(\beta)$, $index(\alpha) \neq$

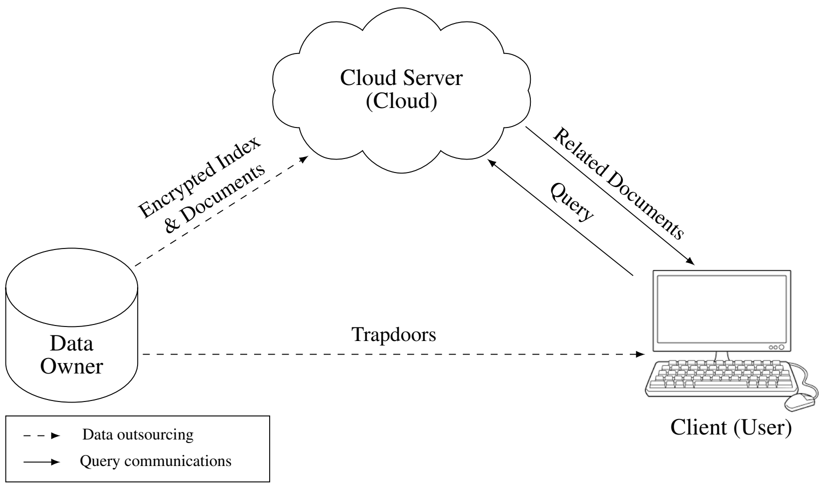

Figure 1: The simplest model of privacy preserving data retrieval system.

$index(\beta)$. Otherwise, the cloud can relate these documents to each other.

2. **No Query Pattern:** For any two documents δ and θ where $features(\delta) = features(\theta)$, $query(\delta,\tau) \neq query(\theta,\tau)$. Otherwise, an unauthorized party can relate the users who are sending similar queries.
3. **No Documents Pattern:** For any unauthorized party, it is infeasible to know the retrieved documents or the rank of the documents for any query $query(\delta,\tau)$. Otherwise, this unauthorized party can relate the retrieved documents to each other, or, relate the query $query(\delta,\tau)$ to the retrieved documents and dissociate it to others.
4. **No Index Frequency:** For any document α, there is no frequency pattern in the index $index(\alpha)$ that can be used to infer any information about the data in that document.
5. **No Query Frequency:** For any document δ and trapdoor τ, there is no frequency pattern in the query $query(\delta,\tau)$ that can be used to infer any information about the data in that document.
6. **No Replay Attack:** For any valid query, it cannot be used later by any unauthorized party for any purpose.
7. **Query Privacy:** Neither the cloud nor any unauthorized party is allowed to know or to be able to infer anything about the contents of the user's queries. Moreover, only authorized users can make queries.
8. **Index Privacy:** For any unauthorized party, it is infeasible to know or to be able to infer anything about the contents of the index. Additionally, in case cloud can add fake indexes (even random ones), it cannot acquire any useful information.
9. **Documents Privacy:** For any unauthorized party, it is infeasible to know or to be able to infer anything about the contents of the encrypted documents.

These 9 security requirements along with with the **high efficiency of data retrieval** are called the 9+1 requirements in the rest of this paper.

One of the first techniques to handle the privacy preserving search on encrypted data was proposed by Song et al. (Song et al., 2000). Later, various techniques were proposed such as (Boneh et al., 2004; Liu et al., 2009; Li et al., 2011; ChinnaSamy and Sujatha, 2012; Kuzu et al., 2012; Tseng et al., 2012). These techniques are examples of keyword-based search techniques. However, this kind of search misses a lot of similarity details and decreases the efficiency of the data retrieval. The techniques in (Li et al., 2010; Chuah and Hu, 2011; Wang et al., 2012b) provided the capability of fuzzy keyword search. These techniques possess the same weaknesses with the keyword-based search techniques. However, they are distinguished by their capability of overcoming a number of spelling mistakes found in the queries. Other techniques such as (Wang et al., 2010; Orencik and Savas, 2014; Wang et al., 2012a; Chen et al., 2014; Sun et al., 2013; Cao et al., 2011) provided results ranking. However, in order to provide results ranking, these techniques compromise some key data to unauthorized parties in the system. In a previous work (Dawoud and Altilar, 2014) we used homomorphic encryption of normalized Term Frequency-Inverse Document Frequency (TF-IDF) values (Rajaraman and Ullman, 2011) to provide a multi-keyword ranked search. The technique was an improvement to Gopal and Singh (Gopal and Singh, 2012) technique to hide any frequency pat-

$$\begin{tikzcd}
 & x \arrow[d] & \boxdot & & y \arrow[d] & = & & z \\
K_H \arrow[r] & \boxed{E} \arrow[d] & & K_H \arrow[r] & \boxed{E} \arrow[d] & & K_H \arrow[r] & \boxed{D} \arrow[u] \\
 & a & \odot & & b & = & & c \arrow[u]
\end{tikzcd}$$

Figure 2: Homomorphic property of homomorphic encryption.

tern in the index as well as keeping the efficiency of data retrieval high. Both techniques utilize the cosine similarity of TF-IDF vectors. The previously proposed technique satisfies the 1st, 4-5th, and 7-9th requirements along with efficiency requirement where it fails to satisfy the 2nd, 3rd, and 6th requirements.

Homomorphic encryption is a form of encryption that allows computations to be applied on the encrypted data. The result is the ciphertext of the value resulting from applying the same operations (or other) on the unencrypted values. Figure 2 shows the homomorphic property of homomorphic encryption. Suppose that x and y are two plaintexts, and, a and b are the ciphertexts resulting from encrypting x and y, respectively, using the same homomorphic key K_H. z is resulting from applying the $\boxdot$ operation on x and y, while c is resulting from applying $\odot$ operation on a and b. $\odot$ and $\boxdot$ severally can be addition, subtraction, multiplication or division operations. The homomorphic property ensures that decrypting c using the key K_H results to z. There are two categories of homomorphic encryption algorithms: partially and fully homomorphic algorithms. Partially homomorphic algorithms support only multiplication or addition, while fully homomorphic algorithms support both multiplication and addition (Xiang et al., 2012).

The rest of this paper is organized as follows: Section 2 discusses the problem statement and the contribution of this paper. Section 3 explains the proposed technique. Section 4 analyse the efficiency of the proposed technique according to the security requirements discussed in this Section. Section 5 concludes this paper and discusses some of the future works.

2 PROBLEM STATEMENT

To the best of our knowledge, there is no technique that satisfies all the 9+1 requirements efficiently in the current state of the art. Failing to satisfy any one of the 9 security requirements poses a threat on the privacy of the data, which is non-negotiable in most of the current applications. On the other hand, high efficiency of data retrieval is needed to achieve the main aim of the data retrieval system. Satisfying some of the security requirements on cost of the retrieval efficiency decreases the reliability of the system.

In a previous work (Dawoud and Altilar, 2014), we proposed a two-rounds technique that uses the model shown in Figure 1. It uses the cosine similarity between the TF-IDF vectors to find a ranked similarity vector of the documents according to a query. This retrieval system was shown more efficient compared to binary keyword-based search systems (Dawoud and Altilar, 2014; Salton and Buckley, 1988a). Although the technique satisfies the requirements 1, 4-5, and 7-9 efficiently, it is still unable to hide the query and document patterns. It is also vulnerable to replay attacks as shown below:

1. **Query Pattern:** The normalization and encryption of the values of a Term Frequency (TF) vector are used to hide any frequency pattern in that vector. However, any two queries with equal TF vectors generate the same normalized values but with different distributions. This is because the normalized values are distributed randomly in place of the original ones. Therefore, this pattern can be detected by checking if any two or more queries have the same values even in different distributions.
2. **Documents Pattern:** The documents are requested in a clear format from the cloud in the second round. Therefore, the cloud can relate the documents requested by a user in the second round to the query sent by the same user in the first round. It can also relate the requested documents to each other.
3. **Replay Attacks:** Any valid query can be reused by any party. Although unauthorized parties cannot compromise the contents of the query, similarity vector, and retrieved documents, they are still able to use these valid queries in Denial-of-Service (DoS) attacks.

Therefore, finding a technique that satisfies all the 9+1 requirements is our contribution in this paper. The technique benefits the achievements of the technique proposed in (Dawoud and Altilar, 2014). It is extended to overcome its deficiencies to reach a complete ranked multi-keyword secure data retrieval system over cloud system.

3 THE TECHNIQUE

Beside the data owner, cloud (which is called searching server in the proposed technique), and user, which are similar to the ones shown in Figure 1, the proposed technique model includes an authentication server, ranking server, private server, and L document servers. Searching server, authentication server, ranking server, private server, and document servers are assumed to be “honest-but-curious”and do not collaborate with each other, which is consistent with previous works. The same assumptions regarding the data owner, cloud, and user reported in Section 1 for the model in Figure 1 are used here. The required authorizations and communication security between the system parties are assumed to be appropriately done. Although hiding the communications paths maybe essential in such systems, it is considered out of the scope of this paper.

Figure 3 shows the model of the proposed technique. It can be divided into six processes: data outsourcing, query generation, query authentication, similarity vector calculation, similarity vector ranking, and documents retrieval. The rest of this Section discusses the implementation of these processes. The security of the proposed technique will be discussed in Section 4.

3.1 Data Outsourcing

The data owner generates the TF-IDF table of the set of documents D (which will be noted as $TFIDF$ in the rest of this paper). The searchable indexes S is generated by normalizing $TFIDF$ values and encrypting them by a homomorphic encryption algorithm and a key K_h as described in (Dawoud and Altilar, 2014). Moreover, the documents (D) and their IDs (ID) are encrypted separately by a symmetric or an asymmetric encryption algorithm and a key K_s to generate $E[D]$ and $E[ID]$, respectively. Thereafter, S is sent to the searching server, K_h and K_s are sent to the user, K_h is sent to the ranking server, $E[ID]$ is sent to the private server, while $E[D]$ and $E[ID]$ are sent to the document servers as shown in Figure 3

3.2 Query Generation

The user calculates the TF vector of the query document (QTF). The values of QTF are normalized as described in (Dawoud and Altilar, 2014) to generate $\Gamma(QTF)$. An extra step before encrypting these normalized values is to multiply each normalized value by a number ρ, which is a random number greater than zero generated for each single query, to generate $\Gamma_\rho(QTF)$. Therefore, if $\Gamma(QTF) = [f_1, f_2, \dots, f_M]$, then $\Gamma_\rho(QTF) = [(f_1 \times \rho), (f_2 \times \rho), \dots, (f_M \times \rho)]$. Multiplication by ρ is used to hide any query pattern as will be shown in Section 4. Finally, the values of $\Gamma_\rho(QTF)$ are encrypted by the same homomorphic encryption algorithm used by the data owner and the key K_h to generate the query vector (Q).

The user sends the query which consists of Q, the number of documents to be retrieved (r), and the authentication key (k_a) (to be discussed in Sub-Section 3.3) to the searching server as shown in Step (1) in Figure 3.

3.3 Query Authentication

Utilizing query anonymous authentication in the proposed technique may provide many properties such as prevention of replay attack, allowing only authorized users to create a valid query, services pricing and billing, privileges granting, etc. However, only the added security properties are discussed in this paper, where the others are considered out of its scope.

In a previous stage of this work, an anonymous authentication technique for limited resources units has been proposed. The system consists of three parties: authentication servers, readers (or foreign servers), and users. The authentication servers have the data needed for authentication. The readers are responsible of transferring data between the users and the servers. The users are limited resources units. Each user has a key K_a which is composed of unique M sub-keys grouped into I groups. Suppose that a combination of sub-keys is a set of sub-keys composed of exactly one sub-key from each group of a key K_a, then, the summation of any combination in the whole system has to be unique. An m different combinations of sub-keys are selected randomly without duplication by the user in each authentication process. The summations of these m combinations of sub-keys are calculated and sent to the server as an authentication key. The server uses a key data to reverse these summations and find the exact combinations used to generate them. The server searches in the database for the user which has all the sets of sub-keys used in the combinations. If such a user is found, the server identify him, otherwise, the request is considered as fake request and ignored.

Suppose that W is the number of users, and $G = \{g_1, g_2, \dots, g_I\}$ is the set of the sizes of the groups. To find the sub-keys that satisfies the above assumptions, the server do the following steps:

1. Find the maximum value of the interval of the sub-keys $[Min, Max]$ as follows:

$$Max = \left(\prod_{i=1}^{I} (W * g_i) + 1 \right) - 1 \qquad (1)$$

2. Use a recursive algorithm to divide each interval into non-overlapped sub-intervals. For example, suppose that $M = 9$, $W = 3$, $I = 3$, and $G = \{3,3,3\}$, then $Max = (10*10*10) - 1 = 999$

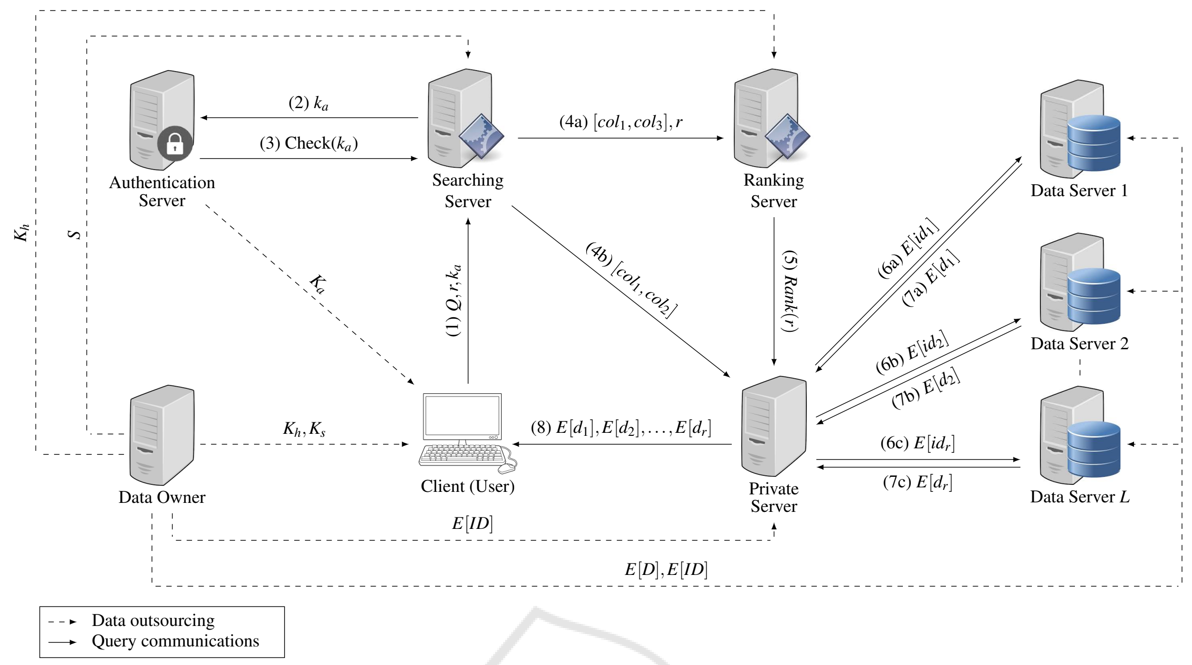

Figure 3: The architecture of the proposed technique.

and the complete interval [Min, 999] is divided into non-overlapped sub-intervals as follows:

$R_1 = \{100, 200, 300, 400, 500, 600, 700, 800, 900\}$

$R_2 = \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$

$R_3 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Note that Min value is 111 and the summation of any combination composed of one value from each interval gives a unique value.

3. Encrypt the values of the sub-intervals individually using the homomorphic encryption and a key K_H to restrict the reversibility of the summations to the parties which has the homomorphic key K_H.
4. Assign sub-keys to the users randomly without duplication from each interval to the related group.

The sub-keys are used to generate different authentication keys, k_a, in each query which is called the authentication key from now on. Therefore, only the authorized users can generate valid authentication keys which are identifiable and acceptable by the authentication server. Moreover, the keys are changing in each authentication process to prevent any unauthorized party (including the readers) from identifying or tracking a user. Although the technique was originally designed to authenticate users with limited resources, it is more than suitable to be applied in the proposed technique based on the following properties:

1. The technique was shown secure against key forgery and key exposure attacks.
2. The technique was shown secure against replay attacks.
3. Generation of an authentication key in the user is very simple (addition of integers).
4. Identification of a user in the server is done by a simple search in the user list without revealing the identity of the user to the reader. In other techniques, the identity of the user is revealed to the reader, or, the server makes a brute-force search to identify a user in each authentication process.
5. Only the authentication server is able to identify the user.
6. No need for key synchronization between the user and the server.
7. After key deployment, the user does not need any data to start an authentication process.
8. The technique is suitable for users with different capabilities starting from simple sensors up to supercomputers.

In the proposed technique, the user and the authentication server play the same roles, while the searching server plays the reader role. The authentication server generates K_a and sends it to the user. In each query, the user generates k_a (which is different for each query) and sends it to the searching server as part of the query. The searching server forwards only k_a to the authentication server. The authentication server checks the validity of k_a. If k_a is valid, the authentication server sends $Accept\_Msg$ to the searching server, otherwise, it sends $Reject\_Msg$ as shown in Steps (2) and (3), respectively, in Figure 3. The searching server checks whether the message coming from the authentication server is $Accept\_Msg$ to proceed, otherwise, the query is ignored.

3.4 Similarity Vector Calculation

In order to find the similarity vector between the query and the documents, the searching server uses the Cosine similarity measure. Cosine similarity measure calculates the cosine value between two vectors (Salton and Buckley, 1988b). Suppose that cs_n is the Cosine similarity between the query $Q = [q_1, q_2, \ldots, q_M]$ and the index of the nth document $s_n = [s_{n,1}, s_{n,2}, \ldots, s_{n,M},]$, then csn can be given as in Equation 2.

$$cs_n = \frac{\sum_{m=1}^{M} (q_m \times s_{n,m})}{\sqrt{\sum_{m=1}^{M} (q_m)^2} \times \sqrt{\sum_{m=1}^{M} (s_{n,m})^2}} \tag{2}$$

The Cosine similarity vector between Q and S (CS) would be the set of all similarity vectors as shown in Equation 3.

$$CS = [cs_n | 1 \leq n \leq N] \tag{3}$$

However, the multiplication of normalized QTF values by ρ in query generation (shown in Sub-Section 3.2) has no effect on the final similarity value. Equation 2 can be reconstructed as in Equation 4 to include this multiplication.

$$cs_n = \frac{\sum_{m=1}^{M} (\rho \times q_m \times s_{n,m})}{\sqrt{\sum_{m=1}^{M} (\rho \times q_m)^2} \times \sqrt{\sum_{m=1}^{M} (s_{n,m})^2}} \tag{4}$$

One can easily simplify Equation 4 to Equation 2 through a single line of derivation as shown in Equation 5.

$$= \frac{\cancel{\rho} \times \sum_{m=1}^{M} (q_m \times s_{n,m})}{\cancel{\sqrt{\rho^2}} \times \sqrt{\sum_{m=1}^{M} (q_m)^2} \times \sqrt{\sum_{m=1}^{M} (s_{n,m})^2}} \tag{5}$$

The searching server creates a table of $N \times 3$ elements where N is the number of documents. The first column (col_1) consists of the numbers between 1 and N distributed randomly in the rows. The second column (col_2) consists of the encrypted IDs ($E[ID]$). The third column (col_3) consists of the cosine similarity values (CS) in the same order of $E[ID]$. The searching server orders the table [col_1, col_2, col_3] according to col_1. Finally, it sends the table [col_1, col_3] and r to the ranking server, while the table [col_1, col_2] is sent to the private server as shown in Steps (4a) and (4b) in Figure 3.

3.5 Similarity Vector Ranking

The ranking server uses K_h to decrypt the values of col_3 received from the searching server in col'_3. The table [col_1, col'_3] is ordered in descending order according to col'_3. The highest r (where r is the number of the requested documents in the query) rows of the ordered [col_1, col'_3] table are stored in a new table called $Rank(r)$. The ranking servers sends $Rank(r)$ to the private server as shown in Step (5) in Figure 3. The private server matches the values of col_1 column of table $Rank(r)$ to the values of col_1 column of the [col_1, col_2] table received from the searching server and retrieves the encrypted IDs of the documents from the column col_2. The matched encrypted IDs (ε) are the encrypted IDs of the documents selected to be retrieved for the query Q.

3.6 Documents Retrieval

Considering that there is L data servers, assume that the private server received υ sets of documents ($\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_\upsilon$) to be retrieved for υ different queries ($Q_1, Q_2, \ldots, Q_\upsilon$), respectively. The private server selects randomly $\kappa \times \sum_{i=1}^{\upsilon} |\varepsilon_i|$ documents from $E[ID]$. These random documents together with the υ sets of documents are inserted randomly in a queue. For each document in the queue, the private server selects a data server randomly to retrieve that document as shown in Steps (6a, 6b, and 6c) in Figure 3. Once a requested document is retrieved from a data server, it is forwarded to the user who sent the query related to that document to decrypt it using K_s as shown in Steps (7a, 7b, and 7c) and (8) in Figure 3, otherwise, it is ignored. The value of κ can be changed according to υ as will be discussed in Section 4.

4 ANALYSIS OF THE TECHNIQUE

This Section discusses the achievement of the 9 security requirements, listed in Section 1, by the proposed technique. Note that the proposed technique relies on a previously proposed technique (Dawoud and Altilar, 2014). The tenth requirement (high efficiency of data retrieval) was discussed in Section 2.

1. **No Index Pattern:** Whatever the values of $TFIDF$ are, normalization described in (Dawoud and Altilar, 2014) guarantees that all the values of $\Gamma(TFIDF)$, and therefore all the values of S, are unique. Uniqueness of S values means that for any two documents α and β where $features(\alpha) = features(\beta)$, $index(\alpha) \neq index(\beta)$, which achieves the first requirement.

2. **No Query Pattern:** For each set of similar values in QTF, normalization generates a new set of unique values and distributes them randomly in place of the original ones. Therefore, the similarity values of different generated queries of a single document are slightly different. As these values are encrypted, then, even small differences in these similarity values will not be detectable by the searching server. To see that, suppose that $H_m = h_1, h_2, \ldots, h_s$ is the histogram of the QTF values of a document d_m. This means that a value x_i in QTF appears h_i times. Therefore, the number of different queries that can be generated from the document d_m is MQ, which is calculated as shown in Equation 6.

$$MQ = \prod_{i=1}^{s} h_i! \qquad \text{for } h_i > 1 \tag{6}$$

   Table 1 shows the average MQ values for three different datasets available in the literature (Hammouda, 2013; Lang, 1995; Volkan, 2012). It can be seen that the probability of generating two similar queries for the same document is less than $\frac{1}{10^{20000}}$, which is negligible. However, these queries can be recognized since they have the same values but in different distributions. For this reason, multiplication by ρ, mentioned in Sub-Section 3.2, is used. Multiplying the values of $\Gamma(QTF)$ by ρ hides the distribution of these values without effecting the final similarity results as shown in Sub-Section 3.4. Therefore, random distribution of the normalized values together with hiding this distribution achieve the second requirement. Moreover, using anonymous authentication makes the searching server unable to identify the user, which is another advantage of using it among the other authentication techniques.

3. **No Documents Pattern:** In (Dawoud and Altilar, 2014), the similarity vector is sent to the user to decrypt it and selects the documents with the highest similarity values. This may cause an extra overhead to the user; it also reveals the IDs of the documents related to the query. In the proposed technique, the ranking server is used to decrypt and order the values, which means that it has the key K_h. To prevent the ranking server from generating queries, query authentication is used as explained in Sub-Section 3.3. As the ranking server does not have the key K_a, it is unable to generate queries that are acceptable by the authentication server.

   The searching server calculates the similarity values of the encrypted indexes without revealing their actual values. However, sending the encrypted similarity vector together with the related document IDs to the ranking server makes it able to reveal the similarity between the query and the exact documents. So, the searching server sends a temporary random IDs (col_1), instead of the original IDs, to the ranking server. In this way, the ranking server is simply decrypting and ordering random numbers before sending them to the private server. On the other hand, the private server uses the information received from both the searching server and the ranking server to find the related documents.

   Referring to Sub-Section 3.6, to retrieve υ sets of documents $(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_\upsilon)$ coming from υ different queries, the private server selects randomly $\kappa \times \sum_{i=1}^{\upsilon} |\varepsilon_i|$ documents from $E[ID]$. These random set of documents together with the υ sets of documents are inserted randomly in a queue. For each document in the queue, the private server selects a data server randomly from L data servers to retrieve that document. Assume that the queue is static, which means that there is no online insertion of documents into the queue. If max is the maximum of $|\varepsilon_1|, \ldots, |\varepsilon_\upsilon|$, then, for a data server l, the probability of being two requested documents related to the same query is $\leq P_r$, where:

$$P_r = \frac{1}{\upsilon(max^2 - max)} \times \frac{1}{((\kappa+1)\sum_{i=1}^{\upsilon} \varepsilon_i)} \times \frac{1}{((\kappa+1)\sum_{i=1}^{\upsilon} \varepsilon_i) - 1} \times \frac{1}{L} \tag{7}$$

   Increasing L decreases P_r, however, finding large number of data servers which are "honest-but-curious"and do not collaborate with each other is not easy. Therefore, for a specific values of L and υ, increasing κ decreases P_r. The private server can dynamically change κ to keep a negligible value of P_r.

4. **No Index Frequency:** Normalization of $TFIDF$ values removes any frequency pattern (Dawoud and Altilar, 2014).

5. **No Query Frequency:** Normalization of QTF

Table 1: Average MQ values of different datasets.

Dataset	Number of Documents	Number of Unique Keywords	Average MQ
webdata (Hammouda, 2013)	314	15756	2033×10^{59270}
mini_newsgroups (Lang, 1995)	400	16360	1115×10^{61691}
classic (Volkan, 2012)	800	6291	3143×10^{21158}

Table 2: Comparison between different privacy-preserving data retrieval techniques.

Requirments	(Boneh et al., 2004)	(Liu et al., 2009)	(Li et al., 2011)	(ChinnaSamy and Sujatha, 2012)	(Kuzu et al., 2012)	(Tseng et al., 2012)	(Li et al., 2010)	(Wang et al., 2012b)	(Chuah and Hu, 2011)	(Wang et al., 2010)	(Cao et al., 2011)	(Wang et al., 2012a)	(Sun et al., 2013)	(Orencik and Savas, 2014)	(Chen et al., 2014)	(Dawoud and Altilar, 2014)	The proposed Technique
1- No Index Pattern	·	·	·	·	√	·	·	·	·	√	·	·	·	·	·	√	√
2- No Query Pattern	·	·	·	·	·	·	·	·	·	√	·	·	·	√	·	·	√
3- No Documents Pattern	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	√
4- No Index Frequency	·	·	·	·	√	·	·	·	·	√	√	·	·	√	√	√	√
5- No Query Frequency	·	·	·	·	√	√	·	·	·	√	√	·	·	√	√	√	√
6- No Replay Attack	·	·	·	·	·	·	·	·	·	·	·	·	·	√	·	·	√
7- Query Privacy	·	·	√	·	√	√	√	√	√	√	√	√	·	√	√	√	√
8- Index Privacy	√	√	√	·	√	√	√	√	√	·	√	·	·	√	√	√	√
9- Documents Privacy	√	√	√	√	√	√	√	√	√	√	√	√	·	√	√	√	√
10- Ranked	·	·	·	·	√	·	·	·	·	√	√	√	√	√	√	√	√

√ = Achieved · = Not achieved.

values removes any frequency pattern (Dawoud and Altilar, 2014).

6. **No Replay Attack:** Each single authentication key k_a is used only once until the authentication keys are updated. Therefore, if a valid query is resent, it will be rejected by the authentication server and ignored by the searching server.

7. **Query Privacy:** The query values are encrypted using K_h. Although the ranking server has the key K_h, it is unable to create a query because it does not have an authentication key. Moreover, it is unable to reveal the query since it does not have the encrypted query. On the other side, the searching server has the encrypted query but does not have the key K_h to decrypt it, which achieves the seventh requirement.

8. **Index Privacy:** The index values are encrypted using K_h. Although the searching server has the encrypted index, it is unable to disclose its values since it does not have the key K_h. Although the ranking server has the key K_h, it also unable to disclose the index contents since it receives only the similarity vector in a random order. Moreover, if the cloud added fake indexes, it will be unable to get any important information since the similarity vector is encrypted. Therefore, the eighth requirement is achieved.

9. **Documents Privacy:** The documents are encrypted using K_s which is known only to the data owner and users. Therefore, unauthorized parties are unable to disclose the contents of the documents. which achieves the ninth property.

Table 2 compares the proposed technique to the techniques reported in the literature according to the 9 security requirements as well as the ranking property. Among the discussed techniques, it can be seen that the only technique that achieves the 9 security requirements is the proposed technique. Moreover, it provides a multi-keyword ranked search based on the similarity of the normalized TF-IDF values which gives better retrieval efficiency compared to the other searching techniques (Dawoud and Altilar, 2014).

5 CONCLUSION

In this paper, 9 security requirements are defined to create a highly secure data retrieval system that utilizes cloud computing systems. These requirements are: no index pattern, no query pattern, no documents pattern, no index frequency, no query frequency, no replay attack, query privacy, index privacy, and documents privacy. None of the techniques that have been reported in the literature are able to satisfy all of these 9 requirements. Moreover, some of the existing approaches use data mining techniques that may decrease the efficiency of the data retrieval, such as binary features, reduction of keywords, reduction of features vector, classes normalization, etc. The proposed technique is shown as being able to satisfy all of the 9 security requirements along with the efficiency requirement. It utilizes a multi-server setting to separate the leaked information. However, none of the servers are able to infer any information from the data that pass through it. The technique uses anonymous authentication of the queries to prevent any unauthorized party from generating a query as well as preventing the replay attacks. It also uses the Cosine similarity measure to calculate the similarity between the TF vector of the query and the TF-IDF vectors of the documents to rank them according to their similarity to the query. This similarity measure is shown as being effective and applicable in the proposed technique. Table 2 illustrates the position of the proposed technique with regard to relevant researches available in the literature. The technique can also be adapted to support fuzzy-keywords retrieval property, which is a future research topic for our group.

REFERENCES

Boneh, D., Di Crescenzo, G., Ostrovsky, R., and Persiano, G. (2004). Public key encryption with keyword search. In Cachin, C. and Camenisch, J., editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer Berlin Heidelberg.

Cao, N., Wang, C., Li, M., Ren, K., and Lou, W. (2011). Privacy-preserving multi-keyword ranked search over encrypted cloud data. In *INFOCOM, 2011 Proceedings IEEE*, pages 829–837.

Chen, L., Sun, X., Xia, Z., and Liu, Q. (2014). An efficient and privacy-preserving semantic multi-keyword ranked search over encrypted cloud data. *International Journal of Security and Its Applications*, 8(2):323–332.

ChinnaSamy, R. and Sujatha, S. (2012). An efficient semantic secure keyword based search scheme in cloud storage services. In *Recent Trends In Information Technology (ICRTIT), 2012 International Conference on*, pages 488–491.

Chuah, M. and Hu, W. (2011). Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data. In *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*, pages 273–281.

Dawoud, M. and Altilar, D. (2014). Privacy-preserving search in data clouds using normalized homomorphic encryption. In *Euro-Par 2014: Parallel Processing Workshops*, volume 8806 of *Lecture Notes in Computer Science*, pages 62–72. Springer International Publishing.

Gopal, G. and Singh, M. (2012). Secure similarity based document retrieval system in cloud. In *Data Science Engineering (ICDSE), 2012 International Conference on*, pages 154–159.

Hammouda, k. (2013). Web mining data - uw-can-dataset. http://pami.uwaterloo.ca/ hammouda/webdata.

Kuzu, M., Islam, M. S., and Kantarcioglu, M. (2012). Efficient similarity search over encrypted data. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 1156–1167, Washington, DC, USA. IEEE Computer Society.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., and Lou, W. (2010). Fuzzy keyword search over encrypted data in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5.

Li, M., Yu, S., Cao, N., and Lou, W. (2011). Authorized private keyword search over encrypted data in cloud computing. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 383–392.

Liu, Q., Wang, G., and Wu, J. (2009). An efficient privacy preserving keyword search scheme in cloud computing. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 2, pages 715–720.

Orencik, C. and Savas, E. (2014). An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking. *Distributed and Parallel Databases*, 32(1):119–160.

Rajaraman, A. and Ullman, J. D. (2011). Data mining. In *Mining of Massive Datasets*, pages 1–17. Cambridge University Press. Cambridge Books Online.

Salton, G. and Buckley, C. (1988a). Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, pages 513–523.

Salton, G. and Buckley, C. (1988b). Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523.

Song, D. X., Wagner, D., and Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55.

Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y. T., and Li, H. (2013). Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, pages 71–82, New York, NY, USA. ACM.

Tseng, F.-K., Liu, Y.-H., and Chen, R.-J. (2012). Toward authenticated and complete query results from cloud storages. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 1204–1209.

Volkan, T. (2012). Data mining research - classic3 and classic4 datasets. http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets.

Wang, C., Cao, N., Li, J., Ren, K., and Lou, W. (2010). Secure ranked keyword search over encrypted cloud data. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 253–262.

Wang, C., Cao, N., Ren, K., and Lou, W. (2012a). Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 23(8):1467–1479.

Wang, C., Ren, K., Yu, S., and Urs, K. (2012b). Achieving usable and privacy-assured similarity search over outsourced cloud data. In *INFOCOM, 2012 Proceedings IEEE*, pages 451–459.

Xiang, G., Yu, B., and Zhu, P. (2012). A algorithm of fully homomorphic encryption. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2030–2033.