A Coherent Privacy-Conserving Ranked Keyword Search Approach in Cloud

K.V.Sowmya, K. Sai Snigdha, M.Susmitha Reddy,Ms.P.S.Latha Kalyampudi,Assistant Professor,Dept.of IT
BVRIT HYDERABAD College of Engineering for Women,Hyderabad

Abstract—*Cloud data owners prefer to outsource documents in an encrypted form for the purpose of privacy preserving. Therefore it is essential to develop efficient and reliable ciphertext search techniques. One challenge is that the relationship between documents will be normally concealed in the process of encryption, which will lead to significant search accuracy performance degradation. Also the volume of data in data centers has experienced a dramatic growth. This will make it even more challenging to design ciphertext search schemes that can provide efficient and reliable online information retrieval on large volume of encrypted data. In this paper, a hierarchical clustering method is proposed to support more search semantics and also to meet the demand for fast ciphertext search within a big data environment. In the search phase, accompanying with the exponential growth of document collection, the search time is reduced to a linear time instead of exponential time. In order to verify the authenticity of search results, a structure called minimum hash sub-tree is designed in this paper. The results show that with a sharp increase of documents in the dataset the search time of the proposed method increases linearly whereas the search time of the traditional method increases exponentially. Furthermore, the proposed method has an advantage over the traditional method in the rank privacy and relevance of retrieved documents.*

Index Terms—Cloud computing, ciphertext search, ranked search, multi-keyword search, hierarchical clustering, security

1.INTRODUCTION

AS we step into the big data era, terabyte of data are produced world-wide per day. Enterprises and users who own a large amount of data usually choose to outsource their precious data to cloud facility in order to reduce data management cost and storage facility spending. As a result, data volume in cloud storage facilities is experiencing a dramatic increase. Although the cloud service providers are ensuring with greater security and privacy, the data owners want to secure their data.

A traditional way to secure data is data encryption. However, this will make server-side data utilization, such as searching on encrypted data, become a very challenging task. In the recent years, researchers have proposed many ciphertext search schemes by incorporating the cryptography techniques. These methods provide greater security, but their methods need massive operations and have high time complexity. Therefore, these methods are not suitable for today's world scenario where data volume is very big and applications require online data processing. In addition, the relationship between documents is concealed in the above methods. The relationship between documents represents the properties of the documents. For example, the relationship can be used to express its category. If a document is independent of any other documents except those documents that are related to hospitals, then it is easy for us to say that this document belongs to the category of hospital. Due to the blind encryption, this important property has been concealed in the traditional methods. Therefore, a new method is introduced which does not conceal the relationship and use it to get faster results.

On the other hand, due to software/hardware failure, and storage corruption, data search results returning to the users may contain damaged data or have been distorted by the malicious administrator or intruder. Thus, a verifiable mechanism should be provided for users to verify the correctness and completeness of the search results.

In this paper, a vector space model is used and every document is represented by a vector. Due to the relationship between different documents, all the

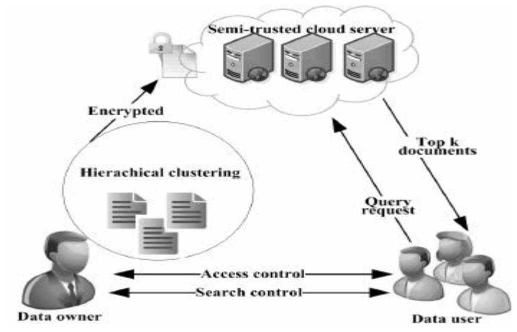


Figure.1.1. Architecture of searching ciphertext

documents can be divided into several categories. Due to this the search time will be reduced to a greater extent as searching will be done only in desired category and dismisses the irrelevant categories. Comparing with all the documents in the dataset, the number of documents which user aims at is very small. Due to the small number of the desired documents, a specific category can be further divided into several sub-categories. Instead of using the traditional sequence search method, a backtracking algorithm is produced to search the target documents. Cloud server will first search the categories and get the minimum desired sub-category. Then the cloud server will select the desired k documents from the minimum desired sub-category. This k value will be decided by the user and will be shared with the cloud previously. If current sub-category cannot satisfy the k documents, cloud server will trace back to its parent and select the desired documents (backtracking). This process will be executed recursively until the desired k documents are satisfied or the root is reached. To verify the integrity of the search result, a verifiable structure based on hash function is constructed. Every document will be hashed and the hash result will be used to represent the document. The hashed results of documents will be hashed again with the category information that these documents belong to and the result will be used to represent the current category. Similarly, every category will be represented by the hash result of the combination of current category information and sub-categories information. A virtual root is constructed to represent all the data and categories. The virtual root is denoted by the hash result of the concatenation of all the categories located in the first level. The virtual root will be signed so that it is verifiable. To verify the search result, user only needs to verify the virtual root, instead of verifying every document.

2 EXISTING SOLUTIONS

2.1 Searchable Encryption with Single Keyword

First introduced the notion of searchable encryption. They propose to encrypt each word in the document independently.

The searching cost in this method is very high as the whole document is to be scanned word by word. Goh [9] formally defined a secure index structure and formulate a security model for index known as semantic security against adaptive chosen keyword attack (ind-cka). They also developed an efficient ind-cka secure index construction called z-idx by using pseudo-random functions and bloom filters. Cash et al. recently design and implement an efficient data structure. Due to the lack of rank mechanism, users have to take a long time to select what they want when massive documents contain the query keyword. Thus, the order-preserving techniques are utilized to realize the rank mechanism, e.g. [10], [11], [12]. Wang et al. [13] use encrypted invert index to achieve secure ranked keyword search over the encrypted documents. In the search phase, the cloud server computes the relevance score between documents and the query. In this way, relevant documents are ranked according to their relevance score and users can get the top-k results. In the public key setting, Boneh et al. [4] designed the first searchable encryption construction, where anyone can use public key to write to the data stored on server but only authorized users owning private key can search. However, all the above mentioned techniques only support single keyword search.

2.2 Searchable Encryption with Multiple Keyword

To enrich search predicates, a variety of conjunctive keyword search methods (e.g. [8], [14]) have been proposed. These methods show large overhead, such as communication cost by sharing secret, or computational cost by bilinear map, e.g.[8]. Pang et al. propose a secure search scheme based on vector space model. Due to the lack of the security analysis for frequency information and practical search performance, it is unclear whether their scheme is secure and efficient or not. Cao et al. present a novel architecture to solve the problem of multi-keyword ranked search over encrypted cloud data. But the search time of this method grows exponentially accompanying with the exponentially increasing size of the document collections. Sun et al. give a new architecture which achieves better search efficiency. However, at the stage of index building process, the relevance between documents is ignored. As a result, the relevance of plaintexts is concealed by the encryption, users expectation cannot be fulfilled well. For example: given a query containing Mobile and Phone, only the documents containing both of the keywords will be retrieved by traditional methods. But if taking the semantic relationship between the documents into consideration, the documents containing Cell and Phone should also be retrieved. Obviously, the second result is better at meeting the users expectation.

3 PROPOSED SOLUTION

In this paper, we propose a **Multi-keyword Ranked Search over Encrypted data based on Hierarchical Clustering Index** (MRSE-HCI) to maintain the close relationship between different plain documents over the encrypted domain in order to enhance the search efficiency. In the proposed architecture, the search time will be increasing linearly even though the data is increasing exponentially. The search time is decreased as the user only concentrates on the desired field rather than searching all the documents. So we can speed up the searching process by computing relevance score between the query and documents which belong to the same specific field with the query. As a result, only documents which are classified to the field specified by users query will be evaluated to get their relevance score. Due to the irrelevant fields ignored, the search speed is enhanced.

We investigate the problem of maintaining the close relationship between different plain documents over an encrypted domain and propose a clustering method to solve this problem. According to the proposed clustering method, every document will be dynamically classified into a specific cluster which has a constraint on the minimum relevance score between different documents in the dataset. This relevance score is used as a measure to evaluate the relationship between different documents. If a new document is added which does not fall under any cluster then the constraint on the cluster may be broken. If one of the new documents breaks the constraint, a new cluster center will be added and the current document will be chosen as a temporal cluster center. Then all the documents will be rearranged and new cluster centers will be selected and according to these new cluster centers the documents will be further divided. Therefore, the number of clusters depends on the number of documents in the dataset and the close relationship between different plain documents. In other words, the cluster centers are created dynamically and the number of clusters is decided by the property of the dataset.

We propose a hierarchical method in order to get a better clustering result within a large amount of data collection. The size of each cluster is controlled as a trade-off between clustering accuracy and query efficiency. According to the proposed method, the number of clusters and the minimum relevance score increase with the increase of the levels whereas the maximum size of a cluster reduces. Every cluster needs to satisfy the constraints. If there is a cluster whose size exceeds the limitation, this cluster will be divided into several sub-clusters.

We design a search strategy to improve the rank privacy. In the search phase, the cloud server will first compute the relevance score between query and cluster centers of the first level and then chooses the nearest cluster. This process will be iterated to get the nearest child cluster until the smallest cluster has been found. The cloud server computes the relevance score between query and documents included in the smallest cluster. If the smallest cluster cannot satisfy the number of desired documents which is previously decided by user, cloud server will trace back to the parent cluster of the smallest cluster and the brother clusters of the smallest cluster will be searched. This process will be iterated until the number of desired documents is satisfied or the root is reached. Due to the special search procedures, the rankings of documents among their search results are different with the rankings derived from traditional sequence search. Therefore, the rank privacy is enhanced.

This authenticated tree structure mainly takes the advantage of the Merkle hash tree and cryptographic signature. Every document will be hashed and the hash result will be used as the representative of the document. The smallest cluster will be represented by the hash result of the combination of the concatenation of the documents included in the smallest cluster and own category information. The parent cluster is represented by the hash result of the combination of the concatenation of its children and own category information. A virtual root is added and represented by the hash result of the concatenation of the categories located in the first level. In addition, the virtual root will be signed so that user can achieve the goal of verifying the search result by verifying the virtual root.

In short, our contributions can be summarized as follows:

1) We investigate the problem of maintaining the close relationship between different plain documents over an encrypted domain and propose a clustering method to solve this problem.
2) We proposed the MRSE-HCI architecture to speed up server-side searching phase. In the search phase, this approach can reach a linear computational complexity against an exponential size increase of document collection
3) We also design a search strategy to improve the rank privacy. In this search strategy backtracking algorithm is used upon the clustering method.
4) We also provide verification mechanism to ensure the correctness and completeness of the results by applying by applying the Merkle hash tree and cryptographic signature to authenticated tree structure.

4 DEFINITION AND BACKGROUND

4.1 System Architecture

The system model contains three entities, the data owner, the data user, and the cloud server.

The data owner is responsible for collecting documents, building document index and outsourcing them in an encrypted format to the cloud server. Apart from that, the data user needs to get the authorization from the data owner before accessing to the data.

The cloud server provides a huge storage space, and the computation resources needed by ciphertext search. Upon receiving a legal request from the data user, the cloud server searches the encrypted index, and sends back top-k documents that are most likely to match users query. The number k is properly chosen by the data user.

In this model, both the data owner and the data user are trusted, while the cloud server is semi-trusted, which is consistent with the architecture in [10]

4.2 Threat Model

The adversaries ability can be concluded in two threat models.

Known ciphertext model: In this model, Cloud server can get encrypted document collection, encrypted data index, and encrypted query keywords.

Known background model: In this model, cloud server knows more information than that in known ciphertext model. Statistical background information of dataset, such as the document frequency and term frequency information of a specific keyword, can be used by the cloud server to launch a statistical attack to infer or identify specific keyword in the query [10], [11], which further reveals the plaintext content of documents. The adversarys ability can be represented in the above two threat models.

4.3 Design Goals Search efficiency.

The time complexity of search time of the MRSE-HCI scheme needs to be logarithmic against the size of data collection in order to deal with the explosive growth of document size in big data scenario. Retrieval accuracy. Retrieval precision is related to two factors: the relevance between the query and the documents in result set, and the relevance of documents in the result set. Integrity of the search result. The integrity of the search results includes three aspects:

1) Correctness. All the documents returned from servers are originally uploaded by the data owner and remain unmodified.
2) Completeness. No qualified documents are omitted from the search results.
3) Freshness. The returned documents are the latest version of documents in the dataset.

Privacy requirements. We set a series of privacy requirements which current researchers mostly focus on.

1) **Data privacy**: Data privacy presents the confidentiality and privacy of documents. Symmetric cryptography is a conventional way to achieve data privacy.
2) **Index privacy:** Index privacy means the ability to frustrate the adversary attempt to steal the information stored in the index. Such information includes keywords and the TF (Term Frequency) of keywords in documents, the topic of documents, and so on.
3) **Keyword privacy:** It is important to protect users query keywords. Secure query generation algorithm should output trapdoors which leak no information about the query keywords.
4) **Rank privacy:** Rank order of search results should be well protected. If the rank order remains unchanged, the adversary can compare the rank order of different search results, further identify the search keyword.

5. ARCHITECTURE & ALGORITHM

5.1 System Model

In this section, we will introduce the MRSE-HCI scheme. The vector space model adopted by the MRSE-HCI scheme is same as the MRSE, while the process of building index is totally different. The hierarchical index structure is introduced into the MRSE-HCI instead of sequence index. In MRSE-HCI, every document is indexed by a vector. Every dimension of the vector stands for a keyword and the value represents whether the keyword appears or not in the document. Similarly, the query is also represented by a vector.

In the search phase, cloud server calculates the relevance score between the query and documents by computing the inner product of the query vector and document vectors and return the target documents to user according to the top k relevance score.

TABEL 5.1
Notations

d_i	The ith document vector, denoted as $d_i = \{d_{i,1}...,d_{i,n}\}$, where $d_{i;j}$ represents whether the j^{th} keyword in the dictionary appears in document d_i.
M	The number of documents in the data collection.
N	The size of dictionary D_W .
CCV	The collection of cluster centers vectors, denoted as CCV=$\{c_1,...,c_n\}$, where c_i is the average vector of all document vectors in the cluster.
CCV_i	The collection of the ith level cluster center vectors, denoted as $CCV_i = \{v_{i;1},...,v_{i;n}\}$ where $V_{i;j}$ represents the jth vector in the ith level.
DC	The information of documents classification such as document id list of a certain cluster.
D_V	The collection of document vectors, denoted as $D_V = \{d_1,d_2,...,d_m\}$.
D_W	The dictionary, denoted as $D_w = \{w_1,w_2,...,w_n\}$.
F_w	The ranked id list of all documents according to their relevance to keyword w.
I_c	The clustering index which contains the encrypted vectors of cluster centers.
I_d	The traditional index which contains encrypted document vectors.
QV	The query vector.
TH	A fixed maximum number of documents in a cluster.
T_w	The encrypted query vector for users query.

computing the inner product of the query vector and document vectors and return the target documents to user according to the top k relevance score.

Due to the fact that all the documents outsourced to the cloud server is encrypted, the semantic relationship between plain documents over the encrypted documents is lost. In order to maintain the semantic relationship between plain documents over the encrypted documents, a clustering method is used to cluster the documents by clustering their related index vectors. Every document vector is viewed as a point in the n-dimensional space. With the length of vectors being normalized, we know that the distance of points in the n-dimensional space reflect the relevance of corresponding documents. In other word, points of high relevant documents are very close to each other in the n-dimensional space. As a result, we can cluster the documents based on the distance measure.

With the volume of data in the data center has experienced a dramatic growth, conventional sequence search approach will be very inefficient. To promote the search efficiency, a hierarchical clustering method is proposed. The proposed hierarchical approach clusters the documents based on the minimum relevance threshold at different levels, and then partitions the resulting clusters into sub-clusters until the constraint on the maximum size of cluster is reached. Upon receiving a legal request, cloud server will search the related indexes layer by layer instead of scanning all indexes.

5.2 MRSE-HCI Architecture

MRSE-HCI architecture is depicted by Figure 4.1 where the data owner builds the encrypted index depending on the

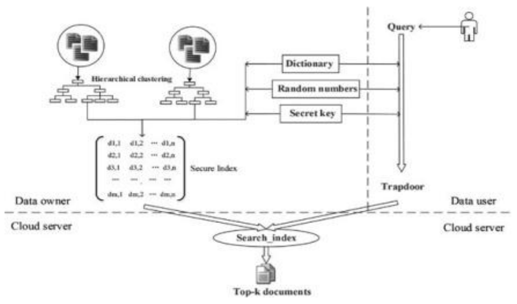


Figure.5.1. MRSE-HCI architecture.

dictionary, random numbers and secret key, the data user submits a query to the cloud server for getting desired documents, and the cloud server returns the target documents to the data user. This architecture mainly consists of following algorithms

$Keygen(1^{l(n)}) \rightarrow (sk, k).$

It is used to generate the secret key to encrypt index and documents.

$Index(D, sk) \rightarrow I.$

Encrypted index is generated in this phase by using the above mentioned secret key. At the same time, clustering process is also included current phase.

$Enc(D, k) \rightarrow E.$

The document collection is encrypted by a symmetric encryption algorithm which achieves semantic security.

$Trapdoor(w, sk) \rightarrow T_w.$

It generates encrypted query vector T_w with users input keywords and secret key.

$Search(T_w, I, k_{top}) \rightarrow (I_w, E_w).$

In this phase, cloud server compares trapdoor with index to get the top-k retrieval results.

$Dec(E_w, k) \rightarrow F_w.$

The returned encrypted documents are decrypted by the key generated in the first step.

Search Index:

The data owner uses tokenizer and parser to analyze every document and gets all keywords. Then data owner uses the dictionary D_w to transform documents to a collection of document vectors DV . Then the data owner calculates the DC and CCV by using a quality hierarchical clustering (QHC) method which will be illustrated in section C. After that, the data owner applies the dimension-expanding and

<u>Algorithm for index.</u>

```
1. input the secret key{sk,k} and data set D
2. extract a directory Dw from D
3. for each document in D
4. costruct a document vector $dv_i$
5. invoke QHC to build hierarchical index {DC,CCV,DV}
6. for every element V in DV and CCV
7.      extend its dimension from n to (n+u+1)
8.      if the $i^{th}$ of S is O
9.      $V'_i = V''_i = V_i$
10.     else
11.       $V'_i$ is set to a random number
12.       $V''_i = V_i - V'_i$
13. encrypt index as $\{M_1{}^T V', M_2{}^T V'', DC\}$
```

<u>Algorithm dynamic k-means.</u>

```
1. input the initial set of k cluster centers C
2. set the threshold $TH_{min}$
3. while k is not stable
   4. generate a new set of cluster centers
            $C_O$ by k-means
5. for every cluster center $C_{o,i}$
6.   get the minimum relevance score: $min(S_i)$
7.      if the $min(S_i) < TH_{min}$
8.          add a new cluster center:k=k+1
9.          go to while
10.Until k is steady
```

vector-splitting procedure to every document vector. It is worth noting that CCV is treated equally as DV . For dimension-expanding, every vector in DV is extended to (n+u+1) bit-long, where the value in n + j(0<=j<=u) dimension is an integer number generated randomly and the last dimension is set to 1. For vector-splitting, every extended document vector is split into two (n+u+1) bitlong vectors, V^{t} and V^{tt} with the help of the ðn þ u þ 1Þbit vector S as a splitting indicator. If the ith element of S (S_i) is 0, then we set

$V_i^{tt} = V_i^{t} = V_i$; If ith element of S (S_i) is 1, then V_i^{tt} is set to a random number and $V_i^{t} = V_i \quad V_i^{tt}$. Finally, the traditional index I_d is encrypted as $I_d = \{M_1^T V^t; M_2^T V^{tt}\}$by using matrix multiplication with the sk, and I_c is generated in a similar way. After this, I_d ,I_c , and DC are outsourced to the cloud server.

$$Search(T_w, I, k_{top}) \rightarrow (I_w, E_w).$$

Upon receiving the T_w from data user, the cloud server computes the relevance score between T_w and index I_c and then chooses the matched cluster which has the highest relevance score. For every document contained in the matched cluster, the cloud server extract its corresponding encrypted document vector in I_d , and calculates its relevance score S with T_w , as described in the Equation (1). Finally, these scores of documents in the matched cluster are sorted and

the top k_{top} documents are returned by the cloud server. The detail will be discussed in the Section 5.5.

$$\begin{aligned} S &= T_w \cdot I_c \\ &= \{M_1^{-1}Q'_w, M_2^{-1}Q''_w\} \cdot \{M_1^T V', M_2^T V''\} \\ &= Q'_w \cdot V' + Q'_w \cdot V'' \\ &= Q_w \cdot V. \end{aligned} \tag{1}$$

5.3 Relevance Measure

In this paper, the concept of coordinate matching is adopted as a relevance measure. It is used to quantify the relevance of document-query and document-document. It is also used to quantify the relevance of the query and cluster centers. Equation (2) defines the relevance score between document d_i and query q_w . Equation (3) defines the relevance score between query q_w and cluster center $lc_{i;j}$. Equation (4) defines the relevance score between document d_i andd_j.

$$S_{qd_i} = \sum_{t=1}^{n+u+1} (q_{w,t} \times d_{i,t}) \tag{2}$$

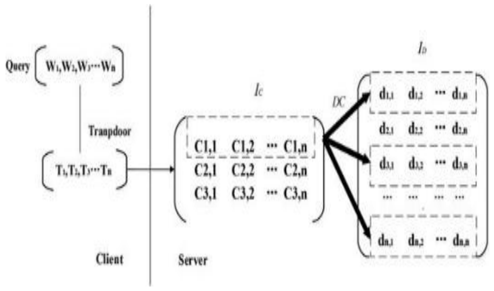
$$S_{qc_i} = \sum_{t=1}^{n+u+1} (q_{w,t} \times lc_{i,j,t}) \tag{3}$$

$$S_{dd_i} = \sum_{t=1}^{n+u+1} (d_{i,t} \times d_{j,t}). \tag{4}$$

5.4 Search Algorithm

The cloud server needs to find the cluster that most matches the query. With the help of cluster index I_c and document classification DC , the cloud server uses an iterative procedure to find the best matched cluster. Following instance demonstrates how to get matched one:

1) The cloud server computes the relevance score between Query T_w and encrypted vectors of the first level cluster centers in cluster index I_c, then chooses the ith cluster center $I_{c;1;i}$ which has the highest score.
2) The cloud server gets the child cluster centers of the cluster center, then computes the relevance score between T_w and every encrypted vectors of child



cluster centers, and finally gets the cluster center $I_{c;2;i}$ with the highest score.

This procedure will be iterated until that the ultimate cluster center $I_{c;l;j}$ in last level l is achieved.

Figure 5.2 : Retrieval Process

In the situation depicted by Fig. , there are nine documents which are grouped into three clusters. After calculating the relevance score with trapdoor T_w , cluster 1, which is shown within the box of dummy line in Fig. 7, is found to be the best match. Documents d_1,d_3 ,d_9 belong to cluster 1, then their encrypted document vectors in the I_d are extracted out to compute the relevance score with T_w.

Algorithm building-minimum hash sub-tree.

```
1.build hash tree based on hierachical clustering result
2.for every leaf node i,
3.  calculate its hash value:
4. while not tree root
5.    for every non-leaf node j
6.    calculate its hash value:
7.    construct node (idj,)
8.    go to upper level
9.calculate tree roots hash value:
10.calculate the signature of hash value
```

5.6 Search Result Verification

The retrieved data have high possibility to be wrong since the network is unstable and the data may be damaged due to the hardware/software failure or malicious administrator or intruder. Verifying the authenticity of search results is emerging as a critical issue in the cloud environment. We, therefore, designed a signed hash tree to verify the correctness and freshness of the search results.

Building. The data owner builds the hash tree based on the hierarchical index structure. The algorithm shown in the Fig. 8 is described as follows. The hash value of the leaf node of the tree is hðid k version k FðidÞÞ where id means document id, version means document version and FðidÞ means the document contents. The value of non-leaf node is a pair of values ðid;hðid k h_{child}ÞÞ where id denotes the value of the cluster center or document vector in the encrypted index, and h_{child} is the hash value of its child node. The hash value of tree root node is based on the hash values of all clusters in the first level. It is worth noting that the root node denotes the data set which contains all clusters. Then the data owner generates the signature of the hash values of the root node and outsources the hash tree including the root signature to the cloud server. Cryptographic signature s (e.g., RSA signature, DSA signature) can be used here to authenticate the hash value of root node.

Processing. By the algorithm shown in the Fig. 9, the cloud server returns the root signature and the minimum hash sub-tree (MHST) to client. The minimum hash sub-tree includes the hash values of leaf nodes in the matched cluster and non-leaf node corresponding to all cluster centers used to find the matched cluster in the searching phase. For example, in the Fig. 10, the search result is document D, E and F. Then the leaf nodes are D, E, F and G, and nonleaf nodes includes C_1, C_2, C_3, C_4, d_D, d_E, d_F , and d_G. In addition, the root is included in the non-leaf node.

Verifying. The data owner uses the minimum hash sub-tree to re-compute the hash values of nodes, in particular the root node which can be further verified by the root signature. If all nodes are matched, then the correctness and freshness is guaranteed. Then the data owner re-searches the index constructed by retrieved values in MHST. If the search result is same as the retrieved result, the completeness, correctness and freshness all are guaranteed.

Algorithm processing-minimum hash sub-tree.

```
 for every leaf node j in the matched cluster
                add its hsh value to the MHST
 change j to its father node inn the upper level
while not tree root
      add j's hash value to the MHST
      for every j's brother node:
            add hash value of the MHST
 change J to its father node in the upper level
 add root signature to the MHST
```

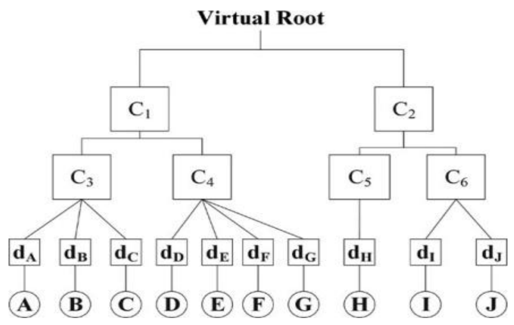


Figure.5.3. Authentication for hierarchical clustering index.

The other values of leaf nodes and non-leaf nodes are generated similarly. In order to combine all first-level clusters into a tree, a virtual root node is created by the data owner with a hash value hðh$_{C1;2}$ k h$_{C2;2}$Þ where $C_{1;2}$ and $C_{2;2}$ denotes the second part of cluster center 1 and 2 respectively. Then the data owner signs the root node and outsources it to the cloud server.

In the processing phase, suppose that the cluster C_4 is the matched cluster and the returned top-three documents are D, E, and F. Then the minimum hash sub-tree includes the hash values of node D, E, F, d_D, d_E, d_F , d_G, C_3, C_2, C_1, C_4 and the signed root .

In the verifying phase, upon receiving the signed root, the data user first check eðhðh$_{C1;2}$ k h$_{C2;2}$Þ;gÞ^{k} ¼$^{?}$ eðsig$_{k}$hðh$_{C1;2}$ k h$_{C2;2}$Þ;gÞ . If it is not true, the retrieved hash tree is not authentic, otherwise the returned nodes, D, E, F, d_D, d_E, d_F , d_G, C_3, C_2, C_1, C_4, works together to verify each other and reconstruct the hash tree. If all the nodes are authenticate, the returned hash tree are authenticate. Then the data user re-computes the hash value of the leaf nodes D, E and F by using returned documents. These new generated hash values are compared with the corresponding returned hash values. If there is no difference, the retrieved documents is correct. Finally, the data user uses the trapdoor to re-search the index constructed by the first part of retrieved nodes. If the search result is same as the retrieved result, the search result is complete.

6 . EFFICIENCY AND SECURITY

6.1 Search Efficiency

The search process can be divided into phase and search(T_w ,I, k_{top})phase. The number of operation needed in phase is illustrated as in Equation (5),where, n is the number of keywords in the dictionary, and w is the number of query keywords,

$$O(MRSE - HCI) = 5n + u - v - w + 5. \qquad (5)$$

Due to the time complexity of Trapdoorðw;skÞ phase independent to DC, when DC increases exponentially, it can be described as O(1).

The difference of the search process between the MRSEHCI and the MRSE is the retrieval algorithm used in this phase. In the SearchðT$_{w}$;I;k$_{top}$Þ phase of the MRSE, the cloud server needs to compute the relevance score between the encrypted query vector T_w and all encrypted document vectors in I_d , and get the top-k ranked document list F_w . The number of operations need in SearchðTW ;I;k_{top}Þ phase is illustrated as in Equation (6), where m represents the number of documents in DC ,and n represents the number of keywords in the dictionary,

$$O(MRSE) = 2m * (2n + 2u + 1) + m - 1. \quad (6)$$

However, in the SearchðTW ;I;k_{top}Þ phase of MRSE-HCI, the cloud server uses the information DC to quickly locate the matched cluster and only compares T_w to a limited number of encrypted document vectors in I_d . The number of operations needed in Search(T_W ,I,$k_{top)}$ phase is illustrated in Equation (7), where k_i represents the number of cluster centers needed to be compared with in the ith level, and c represents the number of document vectors in the matched cluster,

$$O(MRSE - HCI) = \left(\sum_{i=1}^{l} k_i\right) * 2 * (2n + 2u + 1) + c(2 * (2n + 2u + 1)) + c - 1.$$

When DC increases exponentially, m can be set to 2^l. The time complexity of the traditional MRSE is O(2^l) , while the time complexity of the proposed MRSE-HCI is only O(l).

The total search time can be calculated as given in Equation (8) below, where O(trapdoor) is O(1) ,and O(query) relies on the DC,

$$O(searchTime) = O(trapdoor) + O(query). \quad (8)$$

In short, when the number of documents in DC has an exponential growth, the search time of MRSE-HCI increases linearly while the traditional methods increase exponentially

7 CONCLUSION

In this paper, we investigated ciphertext search in the scenario of cloud storage. We explore the problem of maintaining the semantic relationship between different plain documents over the related encrypted documents and give the design method to enhance the performance of the semantic search. We also propose the MRSE-HCI architecture to adapt to the requirements of data explosion, online information retrieval and semantic search. At the same time, a verifiable mechanism is also proposed to guarantee the correctness and completeness of search results.

In addition, we analyze the search efficiency and security under two popular threat models. An experimental platform is built to evaluate the search efficiency, accuracy, and rank security. The experiment result proves that the proposed architecture not only properly solves the multi-keyword ranked search problem, but also brings an improvement in search efficiency, rank security, and the relevance between retrieved documents.

ACKNOWLEDGMENTS

This work was supported by Strategic Priority Research Program of Chinese Academy of Sciences (No. XDA06040601) and Xinjiang Uygur Autonomous Region science and technology plan (No. 201230121). An early version of this paper is presented at the Workshop on Security and Privacy in Big Data at IEEE INFOCOM 2014. Extensive enhancements have been made which includes incorporating a novel verification scheme to help data user verify the authenticity of the search results, and adding a security analysis as well more details of the proposed scheme. This work was supported by Strategic Priority Research Program of Chinese Academy of Sciences (No. XDA06010701) and National High Technology Research and Development Program of China(No. 2013AA01A24).

REFERENCES

[1] Chi Chen, Xiaojie Zhu, Peisong Shen, Jiankun Hu, Song Guo, Zahir Tari, Albert Y. Zomaya," An Efficient Privacy-Preserving Ranked Keyword Search Method"Ieee Transactions On Parallel And Distributed Systems, Vol. 27, No. 4, April 2016,Pp 951-963.

[2] S. Grzonkowski, P. M. Corcoran, and T. Coughlin, "Security analysis of authentication protocols for next-generation mobile and CE cloud services," in Proc. IEEE Int. Conf. Consumer Electron., 2011, Berlin, Germany, 2011, pp. 83–87.

[3] D. X. D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in Proc. IEEE Symp. Security Priv., BERKELEY, CA, 2000, pp. 44–55.

[4] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in Proc. EUROCRYPT, Interlaken, SWITZERLAND, 2004, pp. 506–522.

[5] Y. C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in Proc. 3rd Int. Conf. Applied Cryptography Netw. Security, New York, NY, 2005, pp. 442–455.

[6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in Proc. 13th ACM Conf. Comput. Commun. Security, Alexandria, Virginia, 2006, pp. 79–88.

[7] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in Proc. 27th Annu. Int. Cryptol. Conf. Adv. Cryptol., Santa Barbara, CA, 2007, pp. 535–552.

[8] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in Proc. 4th Conf. Theory Cryptography, Amsterdam, NETHERLANDS, 2007, pp. 535–554.

[9] E.-J. Goh, Secure Indexes, IACR Cryptology ePrint Archive, vol. 2003, pp. 216. 2003.

[10] C. Wang, N. Cao, K. Ren, and W. J. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 8, pp. 1467–1479, Aug. 2012.

[11] A. Swaminathan, Y. Mao, G. M. Su, H. Gou, A. Varna, S. He, M. Wu, and D. Oard, "Confidentiality-preserving rank-ordered search," in Proc. ACM ACM Workshop Storage Security Survivability, Alexandria, VA, 2007, pp. 7–12.

[12] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+R: Topk retrieval from a confidential index," in Proc. 12th Int. Conf. Extending Database Technol.: Adv. Database Technol., Saint Petersburg, Russia, 2009, pp. 439–449.

[13] C. Wang, N. Cao, J. Li, K. Ren, and W. J. Lou, "Secure ranked keyword search over encrypted cloud data," in Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst., Genova, ITALY, 2010, pp. 253–262.

[14] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in Proc. Proc. 2nd Int. Conf. Appl. Cryptography Netw. Security, Yellow Mt, China, 2004, pp. 31–45.