An Efficient Privacy-Preserving Ranked Keyword Search Method

Chi Chen, *Member, IEEE*, Xiaojie Zhu, *Student Member, IEEE*, Peisong Shen, *Student Member, IEEE*, Jiankun Hu, *Member, IEEE*, Song Guo, *Senior Member, IEEE*, Zahir Tari, *Senior Member, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

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. The proposed hierarchical approach clusters the documents based on the minimum relevance threshold, and then partitions the resulting clusters into sub-clusters until the constraint on the maximum size of cluster is reached. In the search phase, this approach can reach a linear computational complexity against an exponential size increase of document collection. In order to verify the authenticity of search results, a structure called minimum hash sub-tree is designed in this paper. Experiments have been conducted using the collection set built from the IEEE Xplore. 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 cloud server providers (CSPs) claim that their cloud service is armed with strong security measures, security and privacy are major obstacles preventing the wider acceptance of cloud computing service [1].

A traditional way to reduce information leakage 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 [34], [35], [36], [37], [43] by incorporating the cryptography techniques. These methods have been proven with provable security, but their methods need massive operations and have high time complexity. Therefore, former methods are not suitable for the big data 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 and hence maintaining the relationship is vital to fully express a document. 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 sports, then it is easy for us to assert this document belongs to the category of the sports. Due to the blind encryption, this important property has been concealed in the traditional methods. Therefore, proposing a method which can maintain and utilize this relationship to speed the search phase is desirable.

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, which means every document can be seen as a point in a high dimensional space. Due to the relationship between different documents, all the

- *C. Chen, X. Zhu and P. Shen is with the State Key Laboratory Of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. E-mail: {chenchi, zhuxiaojie, shenpeisong}@iie.ac.cn.*
- *J. Hu is with the Cyber Security Lab, School of Engineering and IT, University of New South Wales at the Australian Defence Force Academy, Canberra, ACT 2600, Australia. E-mail: J.Hu@adfa.edu.au.*
- *S. Guo is with the School of Computer Science and Engineering, The University of Aizu, Japan. E-mail: sguo@u-aizu.ac.jp.*
- *Z. Tari is with the School of Computer Science, RMIT University, Australia. E-mail: zahir.tari@rmit.edu.au.*
- *A. Y. Zomaya is with the School of Information Technologies, The University of Sydney, Australia. E-mail: albert.zomaya@sydney.edu.au.*

Manuscript received 29 Sept. 2014; revised 8 Apr. 2015; accepted 8 Apr. 2015. Date of publication 21 Apr. 2015; date of current version 16 Mar. 2016. Recommended for acceptance by R. Kwok. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPDS.2015.2425407

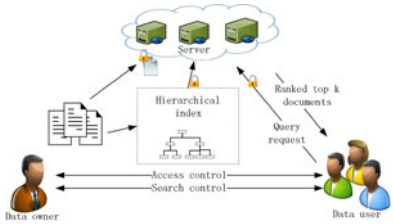


Fig. 1. Architecture of ciphertext search.

documents can be divided into several categories. In other words, the points whose distance are short in the high dimensional space can be classified into a specific category. The search time can be largely reduced by selecting the desired category and abandoning 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. The value of k is previously decided by the user and sent to the cloud server. If current sub-category can not satisfy the k documents, cloud server will trace back to its parent and select the desired documents from its brother categories. 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

In recent years, searchable encryption which provides text search function based on encrypted data has been widely studied, especially in security definition, formalizations and efficiency improvement, e.g. [2], [3], [4], [5], [6], [7]. As shown in Fig. 1, the proposed method is compared with existing solutions and has the advantage in maintaining the relationship between documents.

2.1 Single Keyword Searchable Encryption

Song et al. [2] first introduced the notion of searchable encryption. They propose to encrypt each word in the document independently. This method has a high searching cost due to the scanning of the whole data collection word by word. Goh [8] 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. [41] 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. [9], [10], [11]. Wang et al. [12] 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. [3] 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 Multiple Keyword Searchable Encryption

To enrich search predicates, a variety of conjunctive keyword search methods (e.g. [7], [13], [14], [15], [16]) have been proposed. These methods show large overhead, such as communication cost by sharing secret, e.g. [14], or computational cost by bilinear map, e.g.[7]. Pang et al. [17] 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. [18] 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. [19] 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.

2.3 Verifiable Search Based on Authenticated Index

The idea of data verification has been well studied in the area of databases. In a plaintext database scenario, a variety of methods have been produced, e.g. [20], [21], [22]. Most of these works are based on the original work by Merkle [23], [24] and refinements by Naor and Nissm [25] for certificate revocation. Merkle hash tree and cryptographic signature techniques are used to construct authenticated tree structure upon which end users can verify the correctness and completeness of the query results.

Pang and Mouratidis [26] apply the Merkle hash tree based on authenticated structure to text search engines. However, they only focus on the verification-specific issues ignoring the search privacy preserving capabilities that will be addressed in this paper.

The hash chain is used to construct a single keyword search result verification scheme by Wang et al. [9]. Sun et al. [19] use Merkle hash tree and cryptographic signature to create a verifiable MDB-tree. However, their work cannot be directly used in our architecture which is oriented for privacy-preserving multiple keyword search. Thus, a proper mechanism that can be used to verify the search results within big data scenario is essential to both the CSPs and end users.

3 Our Contribution

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 has a linear growth accompanying with an exponential growing size of data collection. We derive this idea from the observation that users retrieval needs usually concentrate on a specific field. 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. The relevance score is a metric used to evaluate the relationship between different documents. Due to the new documents added to a cluster, 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 reassigned and all the cluster centers will be reelected. 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. Depending on the needs of the grain level, the maximum size of a cluster is set at each level. 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 can not 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.

Some part of the above work has been presented in [27]. For further improvement, we also construct a verifiable tree structure upon the hierarchical clustering method to verify the integrity of the search result in this paper. 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. Accompanying with the exponential growth of document collection, the search time is reduced to a linear time instead of exponential time.
3) We design a search strategy to improve the rank privacy. This search strategy adopts the backtracking algorithm upon the above clustering method. With the growing of the data volume, the advantage of the proposed method in rank privacy tends to be more apparent.
4) By applying the Merkle hash tree and cryptographic signature to authenticated tree structure, we provide a verification mechanism to assure the correctness and completeness of search results.

The organization of the following parts of the paper is as follows: Section 4 describes the system model, threat model, design goals and notations. The architecture and

detailed algorithm are displayed in Section 5. We discuss the efficiency and security of MRSE-HCI scheme in Section 6. An evaluation method is provided in Section 7. Section 8 demonstrates the result of our experiments. Section 9 concludes the paper.

4 Definition and Background

4.1 System Model

The system model contains three entities, as illustrated in Fig. 1, the data owner, the data user, and the cloud server. The box with dashed lines in the figure indicates the added component to the existing architecture.

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 [11]. The number k is properly chosen by the data user. Our system aims at protecting data from leaking information to the cloud server while improving the efficiency of ciphertext search.

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 [9], [18], [28]. In other words, the cloud server will strictly follow the predicated order and try to get more information about the data and the index.

4.2 Threat Model

The adversarys 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 [9], [10], 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. The adversary cannot get the plaintext of documents stored on the cloud server if data privacy is guaranteed. 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) Trapdoor unlinkability. Trapdoor unlinkability means that each trapdoor generated from the query is different, even for the same query. It can be realized by integrating a random function in the trapdoor generation process. If the adversary can deduce the certain set of trapdoors which all corresponds to the same keyword, he can calculate the frequency of this keyword in search request in a certain period. Combined with the document frequency of keyword in known background model, he/she can use statistical attack to identify the plain keyword behind these trapdoors.
  5) 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.

4.4 Notations

In this paper, notations presented in Table 1 are used.

5 Architecture and 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* [18], 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

TABEL 1
Notations

d_i	The ith document vector, denoted as $d_i = \{d_{i,1}, \ldots, d_{i,n}\}$, 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 $CCV = \{c_1, \ldots, c_n\}$, 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 $CCV_i = \{v_{i,1}, \ldots, v_{i,n}\}$ 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 $D_V = \{d_1, d_2, \ldots, d_m\}$.
D_W	The dictionary, denoted as $D_w = \{w_1, w_2, \ldots, w_n\}$.
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.
L_i	The minimum relevance score between different documents in the ith level of a cluster.
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 Fig. 2, where the data owner builds the encrypted index depending on the 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.

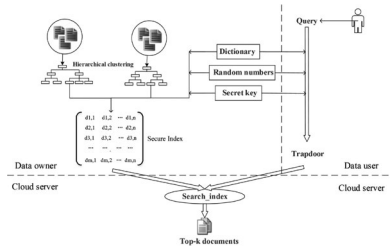


Fig. 2. MRSE-HCI architecture.

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

The concrete functions of different components is described as below.

1) $Keygen(1^{l(n)})$. The data owner randomly generates a $(n + u + 1)$ bit vector S where every element is a integer 1 or 0 and two invertible $(n + u + 1) \times (n + u + 1)$matrices whose elements are random integers as secret key sk. The secret key k is generated by the data owner choosing an n-bit pseudo sequence.
2) $Index(D, sk)$. As show in the Fig. 3, 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

Algorithm Index
1, input the secret key *{sk,k}* and data set *D*
2, extract a dictionary *Dw* from *D*
3, **for** each document in D
4, construct 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 0
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 $\{M_1^T V', M_2^T V'', DC\}$

Fig. 3. Algorithm index.

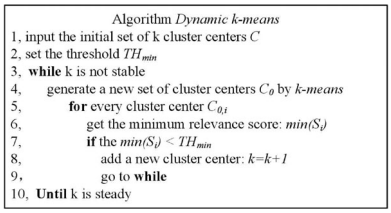

Algorithm *Dynamic k-means*
1, input the initial set of k cluster centers C
2, set the threshold TH_{min}
3, **while** k is not stable
4, generate a new set of cluster centers C_θ by *k-means*
5, **for** every cluster center $C_{\theta i}$
6, get the minimum relevance score: $min(S_i)$
7, **if** the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

Fig. 4. Algorithm dynamic k-means.

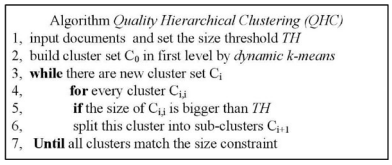

Algorithm *Quality Hierarchical Clustering (QHC)*
1, input documents and set the size threshold TH
2, build cluster set C_0 in first level by *dynamic k-means*
3, **while** there are new cluster set C_i
4, **for** every cluster $C_{i,j}$
5, **if** the size of $C_{i,j}$ is bigger than TH
6, split this cluster into sub-clusters C_{i+1}
7, **Until** all clusters match the size constraint

Fig. 5. Algorithm quality hierarchical clustering (QHC).

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 $n+\ j(0 \le j \le u)$ 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 $(n+u+1)$ bit-long vectors, V' and V'' with the help of the $(n+u+1)$bit vector S as a splitting indicator. If the ith element of S (S_i) is 0, then we set $V_i'' = V_i' = V_i$; If ith element of S (S_i) is 1, then V_i'' is set to a random number and $V_i' = V_i - V_i''$. Finally, the traditional index I_d is encrypted as $I_d = \{M_1^T V', M_2^T V''\}$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.

3) $Enc(D,k)$. The data owner adopts a secure symmetric encryption algorithm (e.g. *AES*) to encrypt the plain document set D and outsources it to the cloud server.

4) $Trapdoor(w,sk)$. The data user sends the query to the data owner who will later analyze the query and builds the query vector QV by analyzing the keywords of query with the help of dictionary D_W, QV then is extended to a $(n+u+1)$ bit query vector. Subsequently,v random positions chosen from a range $(n, n+u]$ in QV are set to 1, others are set to 0.The value at last dimension of QV is set to a random number $t\epsilon[0,1]$. Then the first $(n+u)$dimensions of Q_W , denoted as q_w, is scaled by a random number $r(r \neq 0)$,$Q_w = (r \cdot q_w, t)$. After that, Q_w is split into two random vectors as $\{Q_W', Q_W''\}$ with vector-splitting procedure which is similar to that in the $Index(D,sk)$ phase. The difference is that if the ith bit of S is 1, then we have $q_i' = q_i'' = q_i$; If the ith bit of S is 0, q_i' is set as a random number and $q_i'' = q_i - q_i'$. Finally, the encrypted query vector T_w is generated as $T_w = \{M_1^{-1}Q_w', M_2^{-1}Q_w''\}$ and sent back to the data user.

5) $Search(T_w, I, k_{top})$. 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.

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

6) $Dec(E_w, k)$. The data user utilizes the secret key k to decrypt the returned ciphertext E_w.

5.3 Relevance Measure

In this paper, the concept of coordinate matching [29] 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 and d_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 Quality Hierarchical Clustering Algorithm

So far, a lot of hierarchical clustering methods has been proposed. However all of these methods are not comparable to the partition clustering method in terms of time complexity performance. *K-means* [30] and *K-medois* [31] are popular partition clustering algorithms. But the *k* is fixed in the above two methods, which can not be applied to the situation of dynamic number of cluster centers. We propose a quality hierarchical clustering algorithm based on the novel dynamic *K-means*.

As the proposed *dynamic K-means* algorithm shown in the Fig. 4, the minimum relevance threshold of the clusters is defined to keep the cluster compact and dense. If the relevance score between a document and its center is smaller than the threshold, a new cluster center is added and all the documents are reassigned. The above procedure will be iterated until *k* is stable. Comparing with the traditional clustering method, *k* is dynamically changed during the clustering process. This is why it is called *dynamic K-means* algorithm .

The *QHC* algorithm is illustrated in the Fig. 5. It goes like that. Every cluster will be checked on whether its

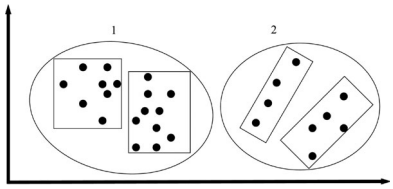


Fig. 6. Clustering process.

size exceeds the maximum number TH or not. If the answer is "yes", this "big" cluster will be split into child clusters which are formed by using the *dynamic K-means* on the documents of this cluster. This procedure will be iterated until all clusters meet the requirement of maximum cluster size. Clustering procedure is illustrated in Fig. 6. All the documents are denoted as points in a coordinate system. These points are initially partitioned into two clusters by using *dynamic K-means* algorithm when the $k = 2$. These two bigger clusters are depicted by the elliptical shape. Then these two clusters are checked to see whether their points satisfy the distance constraint. The second cluster does not meet this requirement, thus a new cluster center is added with $k = 3$ and the *dynamic K-means* algorithm runs again to partition the second cluster into two parts. Then the data owner checks whether these clusters size exceed the maximum number *TH*. Cluster 1 is split into two sub-clusters again due to its big size. Finally all points are clustered into four clusters as depicted by the rectangle.

5.5 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.

In the situation depicted by Fig. 7, 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.

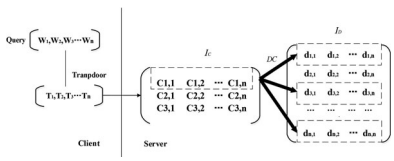


Fig. 7. Retrieval process.

Algorithm *Building-minimum hash sub-tree(MHST)*
1, build hash tree based on hierarchical 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 *(id_p)*
7, go to the upper level
8, calculate tree root's hash value:
9, calculate the signature of hash value

Fig. 8. Algorithm building-minimum hash sub-tree.

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 $h(id \parallel version \parallel \Phi(id))$ where id means document id, $version$ means document version and $\Phi(id)$ means the document contents. The value of non-leaf node is a pair of values $(id, h(id \parallel h_{child}))$ 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 σ (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 non-leaf 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

Algorithm *Processing-minimum hash sub-tree(MHST)*
1, **for** every leaf node *j* in the matched cluster
2, add its hash value to the MHST
3, change *j* to its father node in the upper level
4, **while** not tree root
5, add *j*'s hash value to the MHST
6, **for** every *j*'s brother node:
7, add hash value of to the MHST
8, change *j* to its father node in the upper level
9, add root signature to the MHST

Fig. 9. Algorithm processing-minimum hash sub-tree.

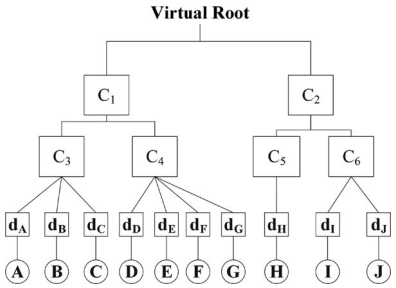


Fig. 10. Authentication for hierarchical clustering index.

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.

As shown in the Fig. 10, in the building phase, all documents are clustered into two big clusters and four small clusters, and each big cluster contains two small clusters. The hash value of leaf node A is $h(id_A \parallel version \parallel \Phi(id_A))$, the value of the non-leaf node C_3 is $(id_{C_3}, h(id_{C_3} \parallel h_A \parallel h_B \parallel h_C))$, and the value of non-leaf node C_1 is $(id_{C_1}, h(id_{C_1} \parallel h_{C_3} \parallel h_{C_4}))$. 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 $h(h_{C_{1,2}} \parallel h_{C_{2,2}})$ 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, e.g., $\sigma(h(h_{C_{1,2}} \parallel h_{C_{2,2}})) = (h_{C_{1,2}} \parallel h_{C_{2,2}}, e(h(h_{C_{1,2}} \parallel h_{C_{2,2}}))^k, g)$, 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 $\sigma(h(h_{C_{1,2}} \parallel h_{C_{2,2}}))$.

In the *verifying* phase, upon receiving the signed root, the data user first check $e(h(h_{C_{1,2}} \parallel h_{C_{2,2}}), g)^k \stackrel{?}{=} e(sig_k h(h_{C_{1,2}} \parallel h_{C_{2,2}}), g)$. 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.

5.7 Dynamic Data Collection

As the documents stored at server may be deleted or modified and new documents may be added to the original data collection, a mechanism which supports dynamic data collection is necessary. A naive way to address these problems is downloading all documents and index locally and updating the data collection and index. However, this method needs huge cost in bandwidth and local storage space.

To avoid updating index frequently, we provide a practical strategy to deal with insertion, deletion and modification operations. Without loss of generality, we use following examples to illustrate the workings of the strategy. The data owner preserves many empty entries in the dictionary for new documents. If a new document contains new keywords, the data owner first adds these new keywords to the dictionary and then constructs a document vector based on the new dictionary. The data owner sends the trapdoor generated by the document vector, encrypted document and encrypted document vector to the cloud sever. The cloud sever finds the closest cluster, and puts the encrypted document and encrypted document vector into it.

As every cluster has a constraint on the maximum size, it is possible that the number of documents in a cluster exceeds the limitation due to the insertion operation. In this case, all the encrypted document vectors belonging to the broken cluster are returned to the data owner. After decryption of the retrieved document vectors, the data owner rebuilds the sub-index based on the deciphered document vectors. The sub-index is re-encrypted and re-outsourced to the cloud server.

Upon receiving a deletion order, the cloud server searches the target document. Then the cloud server deletes the document and the corresponding document vector.

Modifying a document can be described as deleting the old version of the document and inserting the new version. The operation of modifying documents, therefore, can be realized by combining insertion operation and deletion operation.

To deal with this impact on the hash tree, a lazy update strategy is designed. For the insertion operation, the corresponding hash value will be calculated and marked as a raw node, while the original nodes in the hash tree will be kept unchanged because the original hash tree still supports document verification except the new document. Only when the new added document is accessed, the hash tree will be updated. Similar concept is used in the deletion operation. The only difference is that the deletion operation will not bring the hash tree update.

6 Efficiency and Security

6.1 Search Efficiency

The search process can be divided into $Trapdoor(w, sk)$ phase and $Search(T_w, I, k_{top})$ phase. The number of operation needed in $Trapdoor(w, sk)$ 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,

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

Due to the time complexity of $Trapdoor(w, sk)$ phase independent to DC, when DC increases exponentially,it can be described as O(1).

The difference of the search process between the *MRSE-HCI* and the *MRSE* is the retrieval algorithm used in this phase. In the $Search(T_w, I, k_{top})$ 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 $Search(T_W, I, k_{top})$ 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,

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

However, in the $Search(T_W, I, k_{top})$ 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,

$$\begin{aligned} O(MRSE - HCI) = &\left(\sum_{i=1}^{l} k_i\right) * 2 * (2n + 2u + 1) \\ &+ c(2 * (2n + 2u + 1)) + c - 1. \end{aligned} \tag{7}$$

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,

$$O(searchTime) = O(trapdoor) + O(query). \tag{8}$$

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.

6.2 Security Analysis

To express the security analysis briefly, we adopt some concepts from [37], [38], [39] and define what kinds of information will be leaked to the curious-but-honest server.

The basic information of documents and queries are inevitably leaked to the honest-but-curious server since all the data are stored at the server and the queries submitted to the server. Moreover, the access pattern and search pattern cannot be preserved in *MRSE-HCI* as well as previous searchable encryption [18], [38], [39], [40].

Definition 1 (Size pattern). *Let D be a document collection. The size pattern induced by a q-query is a tuple $a(D, Q) = (m, |Q_1|, \ldots, |Q_q|)$ where m is the number of documents and $|Q_i|$ is the size of query Q_i.*

Definition 2 (Access pattern). *Let D be a document collection and I be an index over D. The access pattern induced by a q-query is a tuple $b(D, Q) = (I(Q_1), I(Q_q))$, where $I(Q_i)$ is a set of identifiers returned by query Q_i, for $1 \le i \le q$.*

Definition 3 (Search pattern). *Let D be a document collection. The search pattern induced by a q-query is a $m \times q$ binary matrix $c(D, Q)$ such that for $1 \le i \le m$ and $1 \le j \le q$ the element in the ith row and jth column is 1, if an document identifier id_i is returned by a query Q_j.*

Definition 4 (known ciphertext model secure). *Let $\Pi = (Keygen, Index, Enc, Trapdoor, Search, Dec)$ be an index-based* MRSE-HCI *scheme over dictionary Dw, $n \in N$, be the security parameter, the known ciphertext model secure experiment $PrivK^{kcm}_{A,\Pi}(n)$ is described as follows.*

1) *The adversary submits two document collections D_0 and D_1 with the same length to a challenger.*
2) *The challenger generates a secret key $\{sk, k\}$ by running $Keygen(1^{l(n)})$.*
3) *The challenger randomly choose a bit $b \in \{0, 1\}$, and returns $Index(D_b, sk_b) \rightarrow I_b$ and $Enc(D_b, k_b) \rightarrow E_b$ to the adversary.*
4) *The adversary outputs a bit b'*
5) *The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.*

We say *MRSE-HCI* scheme is secure under known ciphertext model if for all probabilistic polynomial-time adversaries A there exists a negligible function $negl(n)$ such that

$$Pr(Privk^{kcm}_{A,\Pi} = 1) \le 1/2 + negl(n). \tag{9}$$

Proof. The adversary A distinguishes the document collections depending on analyzing the secret key, index and encrypted document collection. Then we have Equation (10), where $Adv(A_D(sk, k))$ is the advantage for adversary A to distinguish the secret key from two random matrixes and two random strings, $Adv(A_D(I))$ is the advantage to distinguish the index from a random string and $Adv(A_D(E))$ is the advantage to distinguish the encrypted documents from random strings.

$$\begin{aligned} &Pr(PrivK^{kcm}_{A,\Pi}(n) = 1) = 1/2 \\ &\quad + Adv(A_D(sk, k)) + Adv(A_D(I)) + Adv(A_D(E)) \end{aligned} \tag{10}$$

□

The elements of two matrixes in the secret key are randomly chosen from $\{0, 1\}^{l(n)}$, and the split indicator S and key k are also chosen uniformly at random from $\{0, 1\}^{l(n)}$. Given $\{0, 1\}^{l(n)}$, A distinguishes the secret key from two random matrixes and two random strings with a negligible probability. Then there exits a negligible function $negl_1(n)$ such that

$$\begin{aligned} Adv(A_D(sk, k)) = &|Pr(Keygen(1^{l(n)}) \rightarrow (sk, k)) \\ &- Pr(Random \rightarrow (sk_r, k_r))| \le negl_1(n), \end{aligned} \tag{11}$$

where sk_r denotes two random matrixes and a random string, and k_r is a random string. In our scheme, the encryption of hierarchical index is essential to encrypt all the document vectors and cluster center vectors. All the cluster center vectors are treated as document vectors in the encryption phase. Eventually, all the document vectors and cluster center vectors are encrypted by the secure KNN. As the secure KNN is known plaintext attack (KPA) secure [32], the hierarchical index is secure under the known ciphertext model. Then there exists a negligible function $negl_2(n)$ satisfying that

$$\begin{aligned} Adv(A_D(I)) = &|Pr(Index(D, sk) \rightarrow (I)) \\ &- Pr(Random \rightarrow (I_r))| \le negl_2(n), \end{aligned} \tag{12}$$

where I_r is a random string.

Since the encryption algorithm used to encrypt D_b is semantic secure, the encrypted documents are secure under

known ciphertext model. Then there exists a negligible function $negl_3(n)$ such that

$$Adv(A_D(E)) = |Pr(Enc(D,k) \rightarrow (E)) - Pr(Random \rightarrow (E_r))| \leq negl_3(n). \quad (13)$$

Where E_r is a random string set.

According Equations (10), (11), (12) and (13), we can get Equation (14),

$$Pr(Privk^{kcm}_{A,\Pi} = 1) \leq 1/2 + negl_1(n) + negl_2(n) + negl_3(n) \quad (14)$$

$$negl(n) = negl_1(n) + negl_2(n) + negl_3(n) \quad (15)$$

$$Pr(Privk^{kcm}_{A,\Pi}) \leq 1/2 + negl(n). \quad (16)$$

By combining Equations (14) and (15), we can conclude Equation (16). Then, we say *MRSE-HCI* is secure under know ciphertext model.

7 Evaluation Method

7.1 Search Precision

The search precision can quantify the users satisfaction. The Retrieval precision is related to two factors: the relevance between documents and the query, and the relevance of documents between each other. Equation (17) defines the relevance between retrieved documents and the query,

$$P_q = \sum_{i=1}^{k'} S(qw, d_i) / \left(\sum_{i=1}^{k} S(qw, d_i) \right). \quad (17)$$

Here, k' denotes the number of files retrieved by the evaluated method, k denotes the number of files retrieved by plain text search, qw represents query vector, d_i represents document vector, and S is a function to compute the relevance score between qw and d_i. Equation (18) defines the relevance of different retrieved documents,

$$P_d = \sum_{j=1}^{k'} \sum_{i=1}^{k'} S(d_j, d_i) / \left(\sum_{j=1}^{k} \sum_{i=1}^{k} S(d_j, d_i) \right). \quad (18)$$

Here, k' denotes the number of files retrieved by the evaluated method, k denotes the number of files retrieved by plaintext search, and both d_i and d_j denote document vector.

Equation (19) combines the relevance between query and retrieved documents and relevance of documents to quantify the search precision such that

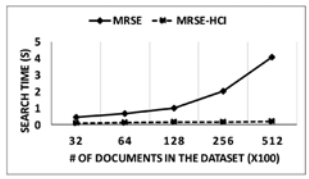
$$Acc = aP_q + P_d, \quad (19)$$

where a functions as a tradeoff parameter to balance the relevance between query and documents and relevance of documents. If a is smaller than 1, it puts more emphasis on the relevance of documents otherwise query keywords.

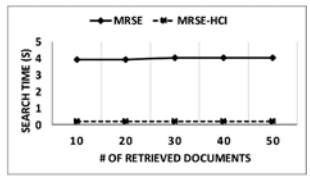
The above evaluation strategies should be based on the same dataset and keywords.

7.2 Rank Privacy

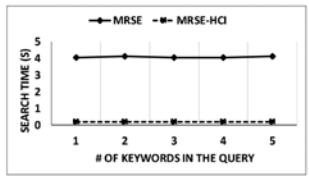
Rank privacy can quantify the information leakage of the search results. The definition of rank privacy is adopted from [18]. Equation (20) is used to evaluate the rank privacy,



(a) Search time with the increasing documents



(b) Search time with the increasing number of retrieved documents



(c) Search time with the increasing number of query keywords

Fig. 11. Search efficiency.

$$P_k = \sum_{i=1}^{k} P_i / k. \quad (20)$$

Here, k denotes the number of top-k retrieved documents, $p_i = |c_{i'} - c_i|$, $c_{i'}$ is the ranking of document d_i in the retrieved top-k documents,c_i is the actual ranking of document d_i in the data set, and P_i is set to k if greater than k . The overall rank privacy measure at point k, denoted as P_k, is defined as the average value of p_i for every document d_i in the retrieved top-k documents.

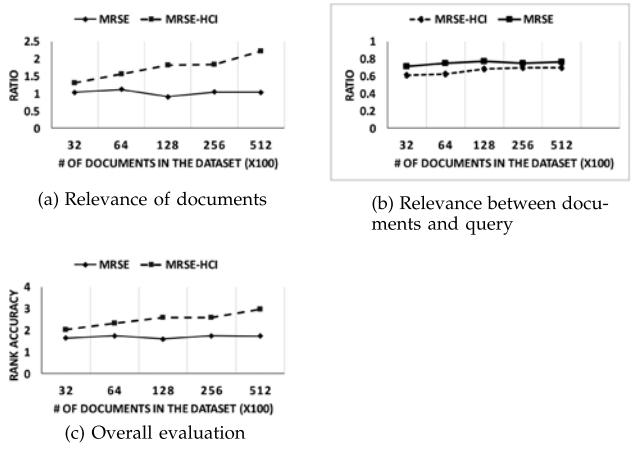
8 Performance Analysis

In order to test the performance of *MRSE-HCI* on real dataset, we built an experimental platform to test the search efficiency, accuracy and rank privacy. We implemented the target experiment based on a distributed platform which includes three *ThinkServer RD830* and a *ThinkCenter M8400t*. The data set is built from *IEEE Xplore*, including about 51,000 documents, and 22,000 keywords.

According to the notations defined in Section 4, n denotes the dictionary size, k denotes the number of top-k documents, m denotes the number of documents in the data set, and w denotes the number of keywords in the users query.

Fig. 11 is used to describe search efficiency with different conditions. Fig. 11a describes search efficiency using the different size of document set with unchanged dictionary size, number of retrieved documents and number of query keywords, $n = 22{,}157, k = 20, w = 5$. In Fig. 11b, we adjust the value of k with unchanged dictionary size, document set size and number of query keywords, $n = 22{,}157$, $m = 51{,}312, w = 5$. Fig. 11c tests the different number of query keywords with unchanged dictionary size, document set size and number of retrieved documents, $n = 22{,}157$, $m = 51{,}312, k = 20$.

From the Fig. 11a, we can observe that with the exponential growth of document set size, the search time of $MRSE$ increases exponentially, while the search time of $MRSE - HCI$ increases linearly. As the Figs. 11b and 11c shows, the search time of $MRSE - HCI$ keeps stable with the increase of query keywords and retrieved documents. Meanwhile, the search time is far below that of *MRSE*.



(a) Relevance of documents

(b) Relevance between documents and query

(c) Overall evaluation

Fig. 12. Search precision.

Fig. 12 describes search accuracy by utilizing *plaintext search* as a standard. Fig. 12a illustrates the relevance of retrieved documents. With the number of documents increases from $3{,}200$ to $51{,}200$, the ratio of *MRSE-to-plaintext search* fluctuates at 1, while *MRSE-HCI-to-plaintext search* increases from 1.5 to 2. From the Fig. 12a, we can observe that the relevance of retrieved documents in the *MRSE-HCI* is almost twice as many as that in the *MRSE*, which means retrieved documents generated by *MRSE-HCI* are much closer to each other. Fig. 12b shows the relevance between query and retrieved documents. With the size of document set increases from $3{,}200$ to $51{,}200$, the *MRSE-to-plaintext search* ratio fluctuates at 0.75. *MRSE-HCI-to-plaintext search* ratio increases from 0.65 to 0.75 accompanying with the growth of document set size. From the Fig. 12b, we can see that the relevance between query and retrieved documents in *MRSE-HCI* is slightly lower than that in *MRSE*. Especially, this gap narrows when the data size increases since a big document data set has a clear category distribution which improves the relevance between query and documents. Fig. 12c shows the rank accuracy according to Equation (19). The tradeoff parameter a is set to 1, which means there is no bias towards relevance of documents or relevance between documents and query. From the result, we can conclude that *MRSE-HCI* is better than *MRSE* in rank accuracy.

Fig. 13 describes the rank privacy according to Equation (20). In this test, no matter the number of retrieved documents, $MRSE - HCI$ has better rank privacy than *MRSE*. This mainly caused by the relevance of documents introduced into search strategy.

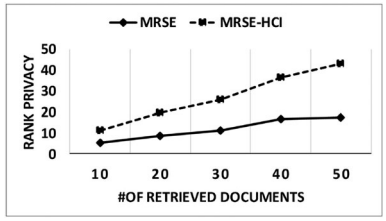


Fig. 13. Rank privacy.

9 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 [27]. 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] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

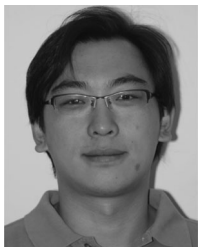
[6] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proc. 27th Annu. Int. Cryptol. Conf. Adv. Cryptol.*, Santa Barbara, CA, 2007, pp. 535–552.

[7] 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.

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

[9] 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.

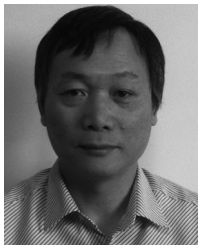
[10] A. Swaminathan, Y. Mao, G. M. Su, H. Gou, A. Varna, S. He, M. Wu, and D. Oard, "Confidentiality-preserving rank-ordered search," in *Proc. ACM ACM Workshop Storage Security Survivability*, Alexandria, VA, 2007, pp. 7–12.
[11] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+R: Top-k retrieval from a confidential index," in *Proc. 12th Int. Conf. Extending Database Technol.: Adv. Database Technol.*, Saint Petersburg, Russia, 2009, pp. 439–449.
[12] 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.
[13] 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.
[14] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. 7th Int. Conf. Inform. Commun. Security*, Beijing, China, 2005, pp. 414–426.
[15] R. Brinkman, "Searching in encrypted data" in University of Twente, PhD thesis, 2007.
[16] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. 1st Int. Conf. Pairing-Based Cryptography*, Tokyo, JAPAN, 2007, pp. 2–22.
[17] H. Pang, J. Shen, and R. Krishnan, "Privacy-preserving similarity-based text retrieval," *ACM Trans. Internet Technol.*, vol. 10, no. 1, pp. 39, Feb. 2010.
[18] N. Cao, C. Wang, M. Li, K. Ren, and W. J. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Shanghai, China, 2011, pp. 829–837.
[19] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. 8th ACM SIGSAC Symp. Inform., Comput. Commun. Security*, Hangzhou, China, 2013, pp. 71–82.
[20] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, "Dynamic authenticated index structures for outsourced databases," in *Proc. ACM SIGMOD*, Chicago, IL, 2006, pp. 121–132.
[21] H. H. Pang and K. L. Tan, "Authenticating query results in edge computing," in *Proc. 20th Int. Conf. Data Eng.*, Boston, MA , 2004, pp. 560–571.
[22] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine, "A general model for authenticated data structures," *Algorithmica*, vol. 39, no. 1, pp. 21–41, May 2004.
[23] C. M. Ralph, "Protocols for public key cryptosystems," in *Proc. IEEE Symp. Security Priv*, Oakland, CA, 1980, pp. 122–122.
[24] R. C. Merkle, "A certified digital signature," in *Proc. Adv. cryptol.* , 1990, vol. 435, pp. 218–238.
[25] M. Naor and K. Nissim, "Certificate revocation and certificate update," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 4, pp. 561–570, Apr. 2000.
[26] H. Pang and K. Mouratidis, "Authenticating the query results of text search engines," in *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 126–137, Aug. 2008.
[27] C. Chen, X. J. Zhu, P. S. Shen, and J. K. Hu, "A hierarchical clustering method For big data oriented ciphertext search," in *Proc. IEEE INFOCOM, Workshop on Security and Privacy in Big Data*, Toronto, Canada, 2014, pp. 559–564.
[28] S. C. Yu, C. Wang, K. Ren, and W. J. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, 2010, pp. 1–9.
[29] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd ed. San Francisco, CA, USA : Morgan Kaufmann, 1999.
[30] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symp. Math. Stat. Prob.*, California, 1967, p. 14.
[31] Z. X. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data Min. Knowl. Discov.*, vol. 2, no. 3, pp. 283–304, Sep. 1998.
[32] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Providence, RI, 2009, pp. 139–152.
[33] R. X. Li, Z. Y. Xu, W. S. Kang, K. C. Yow, and C. Z. Xu, "Efficient Multi-keyword ranked query over encrypted data in cloud computing, *Futur. Gener. Comp. Syst.*, vol. 30, pp. 179–190, Jan. 2014.
[34] G. Craig, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, vol. 9, pp. 169–178
[35] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search[C]," in *Proc. Adv. Cryptol.*, Berlin, Heidelberg, 2004, pp. 506–522.
[36] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean que-ries," in *Proc. Adv. Cryptol,.* Berlin, Heidelberg, 2013, pp. 353–373.
[37] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. Conf. Comput. Commun. Secur.*, 2012, pp. 965–976.
[38] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient construc-tions," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.
[39] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. Adv. Cryptol.*, 2010, pp. 577–594.
[40] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M. C. Rosu, and M. Steiner, "Dynamic searchable encryption in very large databases: Data structures and implementation," in *Proc. Netw. Distrib. Syst. Security Symp.*, vol. 14, 2014, Doi: http://dx.doi.org/10.14722/ndss.2014.23264.
[41] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2013, pp. 875–888.



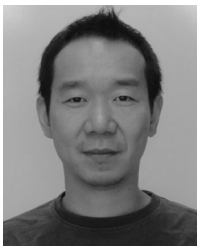
Chi Chen received the BS and MS degrees from Shandong University, Jinan, China, in 2000 and 2003, respectively, and the PHD degree from the Institute of Software Chinese Academy of Sciences, Beijing, China in 2008. He is an associate research fellow of the Institute of Information Engineering, Chinese Academy of Sciences. His research interest includes the cloud security and database security. From 2003 to 2011, he was a research apprentice, research assistant, and associate research fellow with the State Key Laboratory of Information Security, institute of software Chinese Academy of Sciences. Since 2012, he is an associate research fellow with the State Key Laboratory of Information Security, institute of information engineering, Chinese Academy of Sciences, Beijing, China. He is a member of the IEEE.

Xiaojie Zhu received the BS degree in the Zhejiang University of Technology, HangZhou, China, in 2011. He is currently working towards the MS degree in the Institute of Information Engineering, Chinese Academy of Sciences. His research interest includes the information retrieval, secure cloud storage, and data security. He is a student member of the IEEE.

Peisong Shen received the BS degree in the University of Science and Technology of China, HeFei, China, in 2012. He is currently working towards the PhD degree in the Institute of Information Engineering, Chinese Academy of Sciences. His research interest includes the information retrieval, secure cloud storage, and data security. He is a student member of the IEEE.

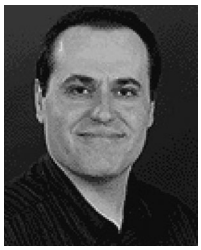


Jiankun Hu is a professor and research director of the Cyber Security Lab, School of Engineering and IT, The University of New South Wales, Canberra, Australia. His main research interest is Cyber security with focus on bio-cryptography, and anomaly intrusion detection. He has obtained seven ARC (Australian Research Council) Grants and has served at the prestigious Panel of mathematics, information and computing sciences, ARC ERA Evaluation Committee. He is a member of the IEEE.



Song Guo received the PhD degree in computer science from University of Ottawa, Canada. He is currently a Full Professor at School of Computer Science and Engineering, the University of Aizu, Japan. His research interests are mainly in the areas of wireless communication and mobile computing, cyber-physical systems, data center networks, cloud computing and networking, big data, and green computing. He has published over 250 papers in referred journals and conferences in these areas and received three IEEE/ACM best paper awards. Dr. Guo currently serves as Secretary of IEEE ComSoc Technical Committee on Satellite and Space Communication (TCSSC) and Technical Subcommittee on Big Data (TSCBD), Associate Editor of IEEE Transactions on Parallel and Distributed Systems (TPDS), IEEE Transactions on Emerging Topics (TETC) for the Computational Networks Track, and on editorial boards of many others. He has also been in organizing and technical committees of numerous international conferences and workshops. Dr. Guo is a senior member of the IEEE and the ACM.

Zahir Tari received the degree in mathematics from the University of Science and Technology Houari Boumediene, Bab-Ezzouar, Algeria, in 1984, the masters degree in operational research from the University of Grenoble, Grenoble, France, in 1985, and the PhD degree in computer science from the University of Grenoble, in 1989. He is a professor, in distributed systems, at RMIT University, Melbourne, Australia. Later, he joined the Database Laboratory at EPFL (Swiss Federal Institute of Technology, 1990-1992) and then moved to QUT (Queensland University of Technology, 1993-1995) and RMIT (Royal Melbourne Institute of Technology, since 1996). He is the head of the DSN (Distributed Systems and Networking) at the School of Computer Scienceand IT, where he pursues high-impact research and development in computer science. He leads a few research groups that focus on some of the core areas, including networking (QoS routing, TCP/IP congestion), distributed systems (performance, security, mobility, reliability), and distributed applications (SCADA, Web/Internet applications, mobile applications).His recent research interests are in performance (in Cloud) and security (in SCADA systems). He regularly publishes in prestigious journals (like *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Trans-actions on Web Services*, *ACM Transactions on Databases*) and conferences (ICDCS, WWW, ICSOC, etc.). He co-authored two books (John Wiley) and edited more than 10 books. He has been the program committee chair of several international conferences, including the DOA (Distributed Object and Appli-cation Symposium), IFIP DS 11.3 on Database Security, and IFIP 2.6 on Data Semantics. He has also been the general chair of more than 12 conferences. He is the recipient of 14 ARC (Australian ResearchCouncil) grants. He is a senior member of the IEEE.



Albert Y. Zomaya is currently the chair professor of high-performance computing and networking and Australian Research Council professorial fellow in the School of Information Technologies, The University of Sydney, Sydney, Australia. He is also the director of the Centre for distributed and high-performance computing which was established in late 2009. He is the author/co-author of seven books, more than 370 papers, and the editor of nine books and 11 conference proceedings. He is the editor in chief of the *IEEE Transactions on Computers* and serves as an associate editor for 19 leading journals. He is the recipient of the Meritorious Service Award in 2000 and the Golden Core Recognition in 2006, both from the IEEE Computer Society. He is a chartered engineer (CEng), a fellow of the AAAS, the IEEE, the IET (UK), and a distinguished engineer of the ACM.