# A Hierarchical Clustering Method For Big Data Oriented Ciphertext Search

Chi Chen, Xiaojie Zhu, Peisong Shen
State Key Laboratory of Information Security,
Institute of Information Engineering, CAS
Beijing, China
{ chenchi, zhuxiaojie, shenpeisong }@iie.ac.cn,

Jiankun Hu
The University of new South Wales,
Canberra, Australia
J.Hu@adfa.edu.au

***Abstract*—Following the wide use of cloud services, the volume of data stored in the data center has experienced a dramatically growth which makes real-time information retrieval much more difficult than before. Furthermore, text information is usually encrypted before being outsourced to data centers in order to protect users' data privacy. Current techniques to search on encrypted data do not perform well within such a massive data environment. In this paper, a hierarchical clustering method for ciphertext search within a big data environment is proposed. The proposed approach clusters the documents based on the minimum similarity threshold, and then partitions the resultant 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 exponential size of document collection. In addition, retrieved documents have a better relationship with each other than traditional methods. An experiment has been conducted using the collection set built from the recent ten years' IEEE INFOCOM publications, including about 3000 documents with nearly 5300 keywords. The results have validated our proposed approach.**

***Index Terms*—cloud computing, ciphertext retrieval, multi-keyword ranked search, hierarchical clustering.**

## I. INTRODUCTION

In the scenario of cloud storage, protecting data privacy is a big challenge for researchers. A traditional way to solve this problem is to encrypt documents before outsourcing documents to clouds. However, this method dramatically degrades the performance of text search on encrypted documents. Many researchers make efforts to solve this problem. Song et al [1] propose an architecture of sequential search over the encrypted documents. Their techniques provide provable secrecy for encryption, in the sense that the untrusted server cannot learn anything about the plaintext given only the ciphertext. Their techniques also support hidden queries, so that the user may ask the untrusted server to search for a secret word without revealing the word to the server. However, the encryption and search algorithms need O(n) number of stream cipher and block cipher operations where *n* represents documents length. This is intolerable against exponential size of data collection within big data environment.

Eu-Jin Goh et al [2] formally define a secure index and formulate a security model for indexes known as semantic security against adaptive chosen keyword attack (ind-cka). They also develop an efficient ind-cka secure index construction called z-idx using pseudo-random functions and Bloom filters. Due to the lack of ranking mechanisms, users have to take a long time to select when massive documents contain the query keyword.

Cong Wang et al [3] use encrypted invert index to achieve secure ranked keyword search over the encrypted documents. In the search phase, the cloud server compute the similarity score between documents and the query. In this way, relevant documents are ranked according to their similarity score and users can get the top *k* results.

All methods mentioned above cannot be directly used in multi-keywords text search. Ning Cao et al [4] present a novel architecture to solve the problem of privacy-preserving multi-keyword ranked search over encrypted cloud data (MRSE). They use coordinate matching to capture the relevance of documents to user's query. Each document is denoted by a binary vector where each bit represents whether corresponding keyword is contained in the document during the index-building phrase. The search query is also described as a binary vector in a similar way. Then they compute relevance score by using matrix operation and a secure inner product computation adapted from k-nearest neighbour(kNN)[9] algorithm.

The search time of the previous methods grows exponentially accompanying with the exponential increase of document collection. Current ciphertext search methods are not suitable to be applied to big data environment where the volume of information in clouds has experienced an explosive growth. As a result, how to efficiently search on ciphertext in the scenario of big data has become a new challenge for researchers. Recently, there is a new trend to solve this problem by using the tree-based search algorithm. Wenhai Sun et al [5] give a new architecture which achieves better search efficiency. However, just as MRSE architecture, the relevance between documents is ignored. The inner similarity between documents and query keywords cannot be fully utilized. As a result, user's expectation cannot be fulfilled. For example: given a query containing "Mobile" and "Phone", the documents only containing both of the keywords will be retrieved by traditional methods. But if taking the relationship between the documents into consideration, the documents containing "Cell" and "Phone" should also be retrieved.



978-1-4799-3088-3/14/$31.00 ©2014 IEEE

Obviously, the second result is better at meeting the user's expectation.

In this paper, we propose a multi-keyword ranked search over encrypted data based on hierarchical clustering index(MRSE-HCI). For the first time, we combine a hierarchical clustering method with MRSE architecture to achieve the goal of real-time information retrieval on encrypted data. In our proposed architecture, the search time has a linear growth accompanying with exponential growing size of data collection. We derive this idea from the observation that user's 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 belonging to this specific field. As a result, only documents which are classified to the field specified by user's query will be evaluated to get their similarity score. It's natural to apply clustering method to getting documents' category information during the index-building phase. At the same time, documents' category information which would be used in the search phase is also needed to be stored in the index.

We propose a novel hierarchical clustering 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. In the query phase, the cloud server will compute the relevance score between query and cluster centers of first level and choose the nearest cluster. We will iterate this procedure to get 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, then returns the most relevant document ID list.

Our contributions are summarized as follows:

1. We apply hierarchical clustering method to our proposed MRSE-HCI architecture. Accompanying with the exponential growth of document collection, the search time is reduced to a linear time instead of exponential time.

2. We firstly introduce a method which evaluates the similarity between documents in the result set. By this way, the retrieved results will focus on the specific field and have a better relevance between documents.

3. Experiments prove that our method is more efficient than MRSE, and the retrieved documents have a closer relationship.

## II. DEFINITION AND BACKGROUND

### A. System Model

In this paper, the system refers to three entities, as illustrated in Fig 1, the data owner, the data user, and the cloud server. The data owner is responsible for collecting documents *F*, building hierarchical clustering index and outsourcing them in an encrypted format to the cloud server. Apart from that, the data owner also needs to manage authorization to the data users. The data user needs to get the authorization from the data owner before accessing to 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 index, and sends back top-k

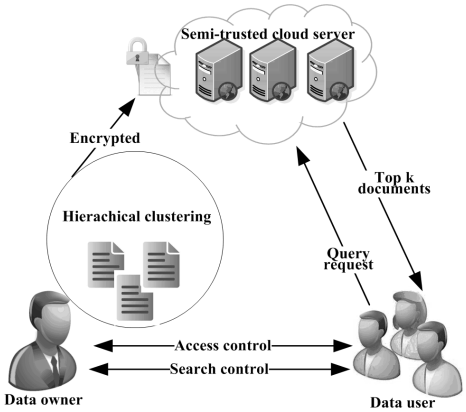



Fig. 1. Architecture of search over outsourced encrypted cloud.

documents that are most likely to match users' intension. [8] The number *k* is properly chosen by the data user. Our system aims at protecting data from leaking any information to the cloud server while improving the efficiency of ciphertext search.

In this paper, both the data owner and the data user are trusted, while the cloud server is semi-trusted, which is consistent with [4] [6] [7]. In other words, the cloud server will strictly follow the predicated order and try to get more information about the data and the index. Under the assumption, there are two popular threat models.

**Known Ciphertext Model**

Cloud server can achieve encrypted document collection, encrypted hierarchical clustering index and encrypted query vector.

**Known Background Model**

Based on the Known Ciphertext Model, cloud server even takes the advantage of statistical and analytic techniques, e.g., the term frequency distribution of a specific keyword. With this information, the cloud server is able to launch statistical attack to infer or identify specific keyword in the query [8-10].

### B. Evaluation

**Retrieval precision evaluation**

The retrieval precision can quantify the users' satisfaction. Retrieval precision is related to two factors: the similarity between documents and the query, and the similarity of documents between each other.

Equation 1 defines the similarity between retrieved documents and the query.

$$P_s = \sum_{i=1}^{k'} S(qw, d_i) / \sum_{i=1}^{k} S(qw, d_i) \quad (1)$$

Where $k'$ denotes the $k$ files retrieved by the evaluated method , $k$ denotes the $k$ files retrieved by plain text search, $q_w$ represents query vector, $d_i$ represents document vector, $S$ is a function to compute the similarity score between $q_w$ and $d_i$ .

Equation 2 defines the relevance of different retrieved documents.

$$P_l = \sum_{j=1}^{k'} \sum_{i>j}^{k'} S(d_j, d_i) / \sum_{j=1}^{k} \sum_{i>j}^{k} S(d_j, d_i) \quad (2)$$

Where $k'$ denotes the $k$ files retrieved by the evaluated method, $k$ denotes the $k$ files retrieved by plain text search, both $d_i$ and $d_j$ denote document vector.

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

**Rank privacy evaluation**

Rank privacy can quantify the information leakage of search results. The definition of "rank privacy" is adopted from [4].

The Equation 3 is used to evaluate the rank privacy.

$$P_k = \sum_{i=1}^{k} P_i / k \tag{3}$$

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

### C. Notations

- $DV$ -the collection of document vectors, denoted as $DV = \{d \mid d_1, d_2, ..., d_m\}$.
- $\mathrm{d}_i$ -the $i^{th}$ document vector, denoted as $\mathrm{d}_i = (\mathrm{d}_{i,1}, \mathrm{d}_{i,2}, ..., \mathrm{d}_{i,n})$ where $d_{i,j}$ represents whether the $j^{th}$ keyword in the dictionary appears in this document $\mathrm{d}_i$.
- $D_w$ -the dictionary, denoted as $D_w = \{w_1, w_2, ..., w_n\}$.
- $CCV$-the collection of cluster center's vectors, denoted as $CCV = \{\mathrm{c}_1, \mathrm{c}_2, ..., \mathrm{c}_1\}$ where $c_i$ is the average vector of all document vectors in the cluster.
- $CCV_i$-the collection of $i^{th}$ level cluster center vectors, denoted as $CCV_i = \{v_{i,1}, v_{i,2}, ...v_{i,x}\}$ where $\mathrm{v}_{i,j}$ represents the $j^{th}$ vector in the $i^{th}$ level.
- $DC$-the information of document's classification such as document id list of a certain cluster.
- $I_c$ -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.
- $T_w$ -the encrypted query vector for user's query.
- $k$ -the number of documents returned back to the user as the search result.
- $F_w$ -the ranked id list of all documents according to their relevance to keyword $w$.
- $TH$ -a fixed maximum number of documents in a cluster.
- $L_i$ -the minimum similarity score between different documents in $i^{th}$ level of a cluster, the higher level corresponds to a larger $L_i$ than lower level in order to contain more documents.
- $u$ -the number of dummy keywords inserted to the document vector.

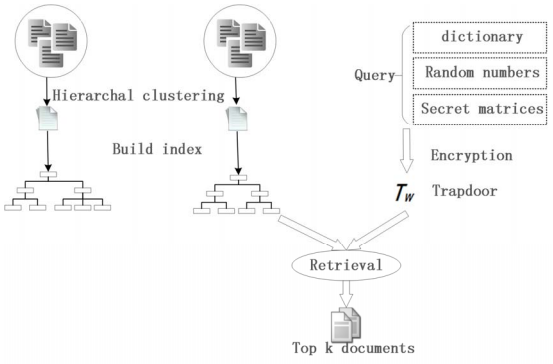



Fig. 2. MRSE-HCI architecture.

- $v$ -the number of chosen keywords from $u$ inserted dummy keywords.

## III. Architecture and Algorithm

### A. MRSE-HCI Architecture

MRSE-HCI architecture consists of the following four parts: (i) $\mathrm{Keygen}(1^{\wedge}\mathrm{n}, \mathrm{u})$ is used to generate the secret key which can be used to encrypt index and documents. (ii) Index is generated in $\mathrm{B}uild-index(\mathrm{F}, \mathrm{SK})$ with the above mentioned secret key. (iii) $\mathrm{T}rapdoor(w, SK)$ generates trapdoor with users' input keywords and secret key. (iv) In $Query(T_w, k, I)$, cloud server uses trapdoor and index to get the top-k retrieval results.

The details of four components are as follows:

- $\mathrm{Keygen}(1^{\wedge}\mathrm{n}, \mathrm{u})$ The data owner randomly generates a $(n+u+1)$ bit vector S and two $(n+u+1)*(n+u+1)$ invertible matrices $\{M_1, M_2\}$ as secret key SK.
- $\mathrm{B}uild-index(\mathrm{F}, \mathrm{SK})$ The client generates a document vector $d_i$ for every document according to $D_W$ and outputs collection of document vectors $DV$. Then the client applies a quality hierarchical clustering ($QHC$) method which outputs the $DC$ and $CCV$. We apply the dimension-expanding and vector-splitting procedure which is similar to the MRSE-II scheme. It's worth noting that $CCV$ is treated equally to $DV$. Every vectors in $DV$ are extended to $(n+u+1)$ bit-long, where the value in $\mathrm{n}+\mathrm{j}(0<j<u+1)$ dimension is a random number and the last dimension is set to 1. Then it's been split to two $(n+u+1)$ bit-long vectors $V'$ and $V''$. Specially, with the $(n+u+1)$ bit vector S as a splitting indicator, if the $i^{th}$ of S ($S_i$) is 0, then set $V'_i = V''_i = V_i$; if $i^{th}$ of S ($S_i$) is 1, $V'$ is set to a random number and $V''_i = V_i - V'_i$. Finally, the traditional index $I_d$ is generated as $I_d = \{M_1^T V', M_2^T V''\}$. The cluster index $I_c$ is generated in the same way. After this, $I_d$ $I_c$ and $DC$ are outsourced to the cloud server.
- $Trapdoor(w, SK)$ The client builds the query vector $QV$ by using dictionary $D_W$, then it's 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. Then the first $(n+u)$ dimensions of $Q_w$, denoted as $q_w$, is scaled by a random number $r(r \neq 0)$, $Q_w = (r * q_w, t)$. Next, $Q_w$ is split into two random vectors as $\{Q_w', Q_w''\}$ with the similar splitting procedure. The difference is that if the $i^{th}$ bit of S is 1, then we have $q_i' = q_i'' = q_i$; If the $i^{th}$ bit of S is 0, $q_i'$ is set a random number and $q_i'' = q_i - q_i'$. Finally, the encrypted trapdoor is $T_w = \{M_1^{-1}Q_w', M_2^{-1}Q_w''\}$.

- $Query(T_w, k, I)$ The data user sends $T_w$ to the cloud server. The cloud server computes the similarity score S between $T_w$ and cluster index $I_d$, then chooses a nearest cluster C with the highest score. The vectors of document included in cluster C will be evaluated by computing the similarity score with the $T_w$. At last, the top $k$ documents with high similarity scores are returned by the cloud server. The detail will be discussed in the next part.

### B. Similarity Measure

In this paper, the "coordinate matching" [9] is adopted as a similarity measure. It is used to quantify the relevance of documents to the query and the other documents. It is also used to quantify the relevance of the query to the cluster centers. Equation 4 is used to define the operation $\oplus$. Equation 5 is used to compute the similarity score between documents $d_i$ and query $QV$. Equation 6 is used to compute the similarity score between $q_w$ and cluster center $CV$. Equation 7 is used to compute the similarity score between document $d_i$ and $d_j$.

$$\oplus = \begin{cases} 1 & 1 \oplus 1 \\ 0 & 1 \oplus 0, 0 \oplus 0, 0 \oplus 1 \end{cases} \quad (4)$$

$$S_i = \sum_{t=1}^{n+u+1} (q_t \oplus d_{i,t}) \quad (5)$$

$$S_i = \sum_{t=1}^{n+u+1} (q_t \oplus v_t) \quad (6)$$

$$S_i = \sum_{t=1}^{n+u+1} (d_{i,t} \oplus d_{j,t}) \quad (7)$$

### C. Hierarchical Clustering Algorithm

Clustering is the task of grouping a set of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters[10]. $K-means$ [11] and $K-medois$ [12] are popular clustering algorithms, but cannot directly used as a hierarchical clustering algorithm. In order to meet the requirements of ciphertext retrieval within big data environment, a quality hierarchical clustering (QHC) algorithm based on dynamic $K-means$ is proposed.

In the proposed QHC algorithm, minimum similarity threshold of the clusters is defined to keep the cluster compact. If the similarity score between a document and its center is smaller than threshold, a new 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 accompanying QHC algorithm uses dynamic $K-means$ algorithm to build

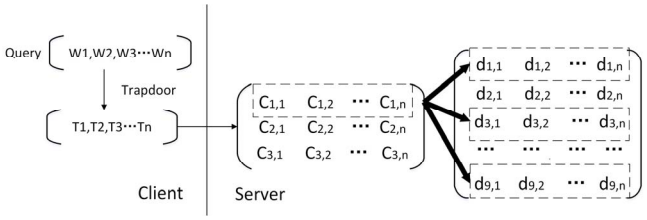



Fig. 3. Retrieval Process.

clusters. Every cluster will be checked whether its size exceeds the maximum number $TH$. If so, this "big" cluster will be split to child clusters which are formed by using dynamic $K-means$ on the documents of this cluster. This procedure will be iterated until all clusters meet the requirement of maximum cluster size.

### D. Retrieval Algorithm

The cloud server searches the encrypted documents by hierarchical clustering index and returns the top-k documents. Following instance demonstrates how to get the ranked top-k documents:

1. The cloud server computes the similarity score between Query $T_w$ and encrypted vectors of the first level cluster centers in cluster index $I_c$, then chooses the $i^{th}$ cluster center $I_{c,1,i}$ which has the highest score.
2. The cloud server gets the child cluster centers of cluster center $I_{c,1,i}$, then compute the similarity score between $T_w$ and every encrypted vectors of child cluster centers, finally get the cluster center $I_{c,2,j}$ with highest score. This procedure will be iterated until the ultimate cluster center $I_{c,l,il}$ in level $l$ is achieved.
3. The cloud server gets the documents which belong to the cluster $I_{c,l,il}$, then the similarity score between the query $Q_w$ and encrypted vectors of these documents in the traditional index $I_d$ is computed. These documents are put into documents list $F_w$. If the number of documents in the ultimate cluster is smaller than k, the documents in the nearest brother are also put into $F_w$.
4. Sort the documents in $F_w$ based on their similarity score and return the top-$k$ documents of $F_w$.

In the situation depicted by fig3, there are 9 docucments which are grouped into 3 clusters. When the trapdoor comes, cluster $1$ within the box of dummy line is found to be the best match. As $d_1, d_3, d_9$ belongs to cluster $1$, then their encrypted vectors in the $I_d$ are taken out to compute the similarity score.

## IV. Efficiency and Security Analysis

### A. Search Efficiency

The search process can be divided into $Trapdoor(w, SK)$ phase and $Query(T_w, k, I)$ phase. The number of operation needed in $Trapdoor(w, SK)$ phase is illustrated as equation 8, where, n is the number of keywords in the dictionary, w is the number of query keywords.

$$MRSE = MRSE - HCI = 5n + u - v - w + 5 \quad (8)$$

When $DC$ increases exponentially, the time complexity of $Trapdoor(w, SK)$ phase is independent to $DC$ and can be described as $O(1)$ .The procedure in $Trapdoor(w, SK)$ phase of MRSE-HCI is the same as MRSE.

The difference of the search process between the MRSE-HCI and the MRSE is the retrieval algorithm used in $Query(T_w, k, I)$ phase.

In the $Query(T_w, k, I)$ phase of MRSE, the cloud server needs to compute the similarity score between the encrypted query vector $T_w$ and all encrypted document vectors $I$ , get the top-k ranked document list $F_w$ . The number of operations need in $Query(T_w, k, I)$ phase is illustrated as equation 9, where m represents the number of documents in $DC$ , n represents the number of keywords in the dictionary.

$$MRSE = 2m * (2n + 2u + 1) + m - 1 \quad (9)$$

However, in the $Query(T_w, k, I)$ phase of MRSE-HCI, the cloud server uses hierarchical clustering index $I$ to search documents. The number of operations needed in $Query(T_w, k, I)$ phase is illustrated as equation 10, where $k_i$ represents the number of cluster centers needed to be compared in the $i^{th}$ level, c represents the number of document vectors in the ultimate cluster.

$$MRSE\text{-HCI} = (\sum_{i=1}^{l} k_i)(2 * (2n + 2u + 1)) + c(2 * (2n + 2u + 1)) + c - 1 \quad (10)$$

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

Total search time can be calculated as equation 11, where $O(trapdoor)$ is $O(1)$ , $O(query)$ rely on the $DC$ .

$$O(searchTime) = O(trapdoor) + O(query) \quad (11)$$

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.

### B. Security Analysis

**Known Ciphertext Model**

The difficulty to infer the plaintext of query vector and document vectors from the hierarchical clustering index and encrypted query vector is determined by the safety strength of secret key, as the hierarchical clustering index and query vector are encrypted before being outsourced to the cloud. This has been proven in the known ciphertext model in [13].

**Know Background Model**

In this model, cloud server will record the search results and analyze the search process, infer the relationship between different $Trapdoor(w, SK)$ . Known the relationship between trapdoors, cloud server cannot get plaintext to identify the keywords. The difficulty for cloud server to get the query keywords is similar to the difficulty of cracking the algorithm.

*Query Unlinkability* Using random number can generate different $Trapdoor(w, SK)$ with same keywords. First, v bit are randomly chosen from u bit. Second, there is a random number r as a scale factor. Third, the value in the last dimension of query vector is a random number.

*Keyword Privacy*. As MRSE-II scheme which is integrated into proposed MRSE-HCI scheme has been proven to be secure in keyword privacy, our proposed scheme also achieve good keyword privacy.

By setting a proper $u$ , users can confuse the relationship between the query and search results, increase the difficulty for cloud server to infer the keywords by using statistical attack. But, a bigger $u$ will degrade the accuracy of query, it is a trade-off between keywords privacy and accuracy of query.

## V. Performance Analysis

In order to test the performance of MRSE-HCI on real dataset, we built an experimental platform to test the search efficiency and accuracy. We implement the platform by JAVA on a Linux server with Intel Core i7 3.40GHZ. The collection set is built from the recent ten years' IEEE INFOCOM publications, including about 3000 publications, from which we extract nearly 5300 keywords.

Fig.4 is used to describe search efficiency with different conditions. Fig.4(a) describes search efficiency using the different size of document set with the same dictionary and keywords, n=5300, k=30, w=5. In Fig.4(b), we adjust the different value of *k* with the same document set and keywords, n=5300, m=3000, w=5. Fig.4(c) test the different number of keywords with the same document set and equal number of retrieved documents, m=3000, k=30. Fig.4(d) uses the different size of dictionary with the same document set and keywords, m=3000, k=30, w=5.

Fig.5 is used to describe the similarity with different conditions. In Fig.5(a), we test the similarity between documents for the different size of document set with the same dictionary and number of retrieved documents, n=5300, k=30, w=5. Fig.5(b) describes the similarity between documents and query for the different size of document set with the same dictionary and number of retrieved documents, n=5300, k=30, w=5. Data shows that the similarity between documents in search results increases dramatically, while the similarity between documents and the query decreases within acceptable

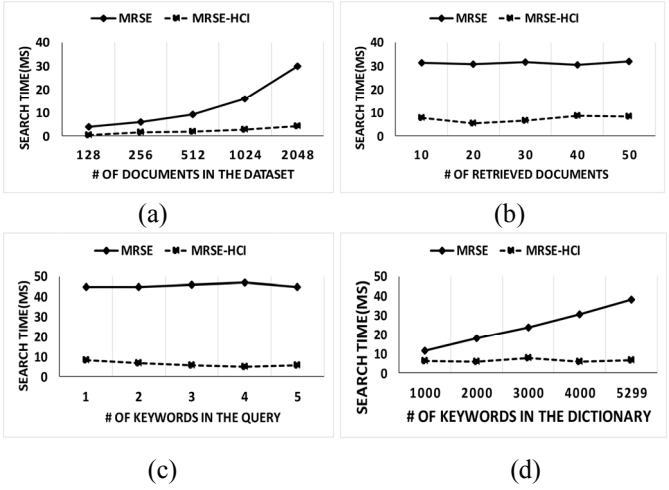
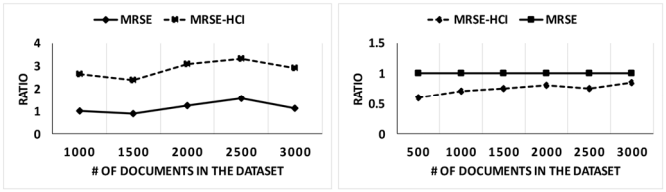



Fig. 4. Search efficiency.




(a) (b)

Fig. 5. Search precision based on plaintext search.

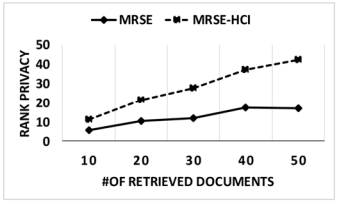



Fig. 6. Rank privacy.

range.

Fig. 6 describes the rank privacy using different number of retrieved documents, with the same dictionary and document set n=5300, m=3000, w=5. Test result shows that the rank privacy is improved.

Based on the above experiments, the proposed MRSE-HCI has an advantage in search efficiency, rank privacy and the similarity between documents in the result set than the traditional MRSE. We also find that MRSE only search the documents which is directly related to the query, but MRSE-HCI can search the documents which is indirectly related with the query. In addition, The MRSE-HCI can also combine with following methods to further improve search efficiency. First, by building proper invertible matrices, we can divide the matrices into many parts, then get the useful parts and abandon the unnecessary parts, which is similar to [14]. Second, we can use other multi-keyword semantics (e.g., weighted query) over encrypted data.

## VI. CONCLUSION

In this paper, we introduce the similarity between different documents into ciphertext search. We also define the MRSE-HCI architecture and related algorithm. 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 similarity between retrieved documents.

## VII. ACKNOWLEDGMENT

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] Dawn Xiaoding Song, David Wagner, and Adrian Perrig, “Practical Techniques for Searches on Encrypted Data”, in Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on (IEEE, 2000), pp. 44-55.

[2] Eu-Jin Goh, "Secure Indexes", IACR Cryptology ePrint Archive, 2003 (2003), 216

[3] Cong Wang, Ning Cao, Jin Li, Kui Ren, and Wenjing Lou, “Secure Ranked Keyword Search over Encrypted Cloud Data”, in Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on (IEEE, 2010), pp. 253-62

[4] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou, “Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data”, in INFOCOM, 2011 Proceedings IEEE (IEEE, 2011), pp. 829-37..

[5] Wenhai Sun, Bing Wang, Ning Cao, Ming Li, Wenjing Lou, Y Thomas Hou, and Hui Li, “Privacy-Preserving Multi-Keyword Text Search in the Cloud Supporting Similarity-Based Ranking”, in Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security (ACM, 2013), pp. 71-82.

[6] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou, “Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing”, in INFOCOM, 2010 Proceedings IEEE (IEEE, 2010), pp. 1-9

[7] Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou, “Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data”, IEEE Transactions on Parallel and Distributed Systems, 23 (2012), 1467-79..

[8] Sergej Zerr, Daniel Olmedilla, Wolfgang Nejdl, and Wolf Siberski, “Zerber+ R: Top-K Retrieval from a Confidential Index”, in Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (ACM, 2009), pp. 439-49.

[9] William Hersh, “Managing Gigabytes—Compressing and Indexing Documents and Images”, Information Retrieval, 4 (2001), 79-80.

[10] WIKIPEDIA,"Cluster analysis".

[11] James MacQueen, “Some Methods for Classification and Analysis of Multivariate Observations”, in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (California, USA, 1967), p. 14.

[12] Zhexue Huang, “Extensions to the K-Means Algorithm for Clustering Large Data Sets with Categorical Values”, Data Mining and Knowledge Discovery, 2 (1998), 283-304.

[13] Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis, “Secure Knn Computation on Encrypted Databases”, in Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (ACM, 2009), pp. 139-52.

[14] Ruixuan Li, Zhiyong Xu, Wanshang Kang, Kin Choong Yow, and Cheng-Zhong Xu, “Efficient Multi-Keyword Ranked Query over Encrypted Data in Cloud Computing”, Future Generation Computer Systems (2013).