Soft Computing (2018) 22:3111–3121
https://doi.org/10.1007/s00500-018-3145-6

METHODOLOGIES AND APPLICATION

Improving file locality in multi-keyword top-k search based on clustering

Lanxiang Chen[1] · Nan Zhang[1] · Kuan-Ching Li[2] · Shuibing He[3] · Linbing Qiu[1]

Published online: 21 March 2018

Abstract
Nowadays, fast growing number of users and business are motivated to outsource their private data to public cloud servers. Taking into consideration security issues, private data should be encrypted before being outsourced to remote servers, though this makes traditional plaintext keyword search rather difficult. For this reason, there exists an urgent need of an efficient and secure searchable encryption technology. In this paper, an affinity propagation (AP) K-means clustering method (CAK-means, a combination of AP and K-means clustering) is proposed to realize fast searchable encryption in Big Data environments. CAK-means clustering utilizes affinity propagation to initialize K-means clustering, thereby making the clustering process faster, stable and effectively improving the initial clustering center quality of the K-means. As the AP algorithm identifies the clustering center with much lower errors than other methods, it significantly improves the search accuracy. Simultaneously, the related files in one cluster are stored at the contiguous locality of disks which will substantially improve the file locality and speedup the read and write disk I/O. Additionally, the coordinated matching measure is utilized to support accurate ranking of search results. Experimental results show that the proposed CAK-means-based multi-keyword ranked searchable encryption scheme (MRSE-CAK) has higher search efficiency and accuracy while simultaneously ensuring equivalent security.

Keywords Searchable symmetric encryption · CAK-means clustering · File locality · Multi-keyword · Ranked search

1 Introduction

Outsourcing data to public cloud are a major trend that offers great benefits to data owners, so rapid increasing number of enterprises and individual users store and share their data in cloud-based storage servers. Because outsourcing data raise confidentiality and privacy concerns, private data need to be encrypted before being outsourced to remote cloud servers. However, this makes traditional plaintext keyword search rather difficult. Recently, a number of searchable symmetric encryption (SSE) schemes (Chen et al. 2016; Cao et al. 2014; Wang et al. 2014; Chen et al. 2017; Xia et al. 2016; Chen et al. 2018; Wang et al. 2017) with multi-functions have been proposed by utilizing diverse cryptography mechanisms.

Considering to the fact that files belonging to the same category have certain relevant characteristics representing their properties, it is essential to maintain the relationships between the related files and full description of each file that may be category or attribute. As user searches the files, he often searches certain contents in portions of data, such as data about "affinity propagation," or certain type files, such as video files or text files. There must be some explicit or implicit relationship between these searched files. Due to data encryption, these important properties may be concealed. Nevertheless, it is vital to keep and utilize these properties to speed up the search efficiency and accuracy in Big Data.

As Poh et al. (2017) pointed out, barely all existing SSE schemes concern only about the search time, the overhead of reading these identifiers of queried files in the disks of storage servers is not analyzed, which is the issue of locality in SSE schemes. In the inverted index-based scheme, a query based on a keyword directly returns the r matching file identifiers.

Communicated by V. Loia.

✉ Lanxiang Chen
lxiangchen@fjnu.edu.cn

[1] Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China

[2] Department of Computer Science and Information Engineering (CSIE), Providence University, Taichung, Taiwan

[3] Computer School, Wuhan University, Wuhan 430072, China

The asymptotic search performance is known to be optimal (O(r)). However, as discussed by Cash and Tessaro (2014), access latency of the r identifiers is not factored in. In practice, these identifiers are stored in pseudo-random locations on storage hardware, implying that retrieval of each identifier requires a read access. On the other hand, these identifiers are stored in contiguous sectors that can be read simultaneously in a plaintext search index.

It is believed that the issue of locality not only affects the read access of files identifiers, but also concerns about the overhead of reading files from the disks of storage servers. The overhead of read access to files identifiers is closely related to index structures. Yet, the overhead of reading the files is closely related to the locality of the files. Maintaining the relationship among related files can be used to solve the problem of locality and speedup the read and write disk I/O. Since the files in one cluster are usually the files needed to be accessed concurrently, they are stored at contiguous disk locations, which will improve the file locality as well as the latency of disk I/O.

Based on the relationship between the files, which is viewed as a classification processing, large amount of data in the cloud can be searched by category, which can be used to improve the search efficiency and accuracy. It can also be used to greatly improve read and write disk I/O. As Chen et al. (2016) stated previously, it is easier to take into consideration the semantic relationship between the files if one utilizes clustering algorithm.

In this paper, MRSE-CAK, a multi-keyword top-k search over encrypted data based on CAK-means (Zhu et al. 2009) is proposed. It focuses on the optimization of the clustering algorithm for higher search accuracy and efficiency, by utilizing the file locality to improve the performance of the read and write disk I/O at the same time. As for the validation of search results and dynamic update of data, existing methods (Chen et al. 2016) are referenced. As overall, the contributions of this work are summarized as:

- The proposed scheme is designed to achieve higher search efficiency and search precision than the MRSE-HCI scheme (Chen et al. 2016) through the usage of the CAK-means algorithm, which not only makes the process of clustering faster and more stable, but also effectively improves the initial clustering center quality of K-means,
- In order to maintain the property relationships among plaintext files, the clustering method is applied. On the one hand, it can achieve semantic search and improve the search accuracy. On the other hand, related files are stored at contiguous locations of cloud servers, and this can speed up the disk I/O and thereby improve the search efficiency of cloud servers,
- Although related files are stored at contiguous locations of cloud servers, security is not weakened in such a design. As user issues keyword queries, the server knows exactly which encrypted files satisfy this query, and it does not matter whether the files are stored dispersedly or continuously on the server,
- To improve search accuracy, the keyword weights (term frequency) are added to index vectors in the MRSE-CAK. The search time of the proposed scheme achieves logarithmic time complexity that is suitable for large data application scenarios,
- The proposed scheme supports multi-keyword search and provides similarity ranking of search results, showing same level of security as the MRSE-HCI scheme. From the design and construction of this scheme, we demonstrate that such a scheme can maintain, or even improve, data privacy, index privacy, keyword privacy, and rank privacy.

The remainder of this paper is organized as follows. Section 2 presents the related works, and Sect. 3 describes notations and preliminaries. Section 4 details the MRSE-CAK construction. Section 5 gives the security and performance analysis, and finally, Sect. 6 concludes the paper and discusses about future directions of this work.

2 Related work

To improve the accuracy of search results, a variety of multi-keyword search methods have been proposed. Cao et al. (2014) proposed a privacy-preserving multi-keyword ranked searchable encryption (MRSE) scheme, where similarity measure of "coordinate matching" and "inner product similarity" was incorporated to quantitatively evaluate this measure. This approach can return the ranked results of the search based on the number of matching keywords. However, MRSE does not take the access frequencies of keywords into account. Wang et al. (2014) propose a privacy-preserving multi-keyword fuzzy search scheme over encrypted data in the cloud, in which the file index is built using the locality-sensitive hashing (LSH) function in the Bloom filter, providing a well-organized solution to the secure fuzzy keyword search.

Chen et al. (2017) propose a dynamic multi-keyword ranked search (DMRS) scheme that makes use of the sparse matrix to replace the dense large-scale matrix of MRSE Cao et al. (2014) in index encryption and query vector encryption to improve the efficiency. The proposed scheme incorporates "coordinate matching" and "inner product similarity" to improve the relevance of search keywords to the relevant cloud files. They also use a reverse data structure to allow users to perform dynamic operations on document collection. Xia et al. (2016) propose a "Greedy Depth-first Search" algorithm based on a tree-based index structure to provide efficient multi-keyword ranked search. It combined the TF $\times$ IDF model and the secure kNN algorithm to achieve sub-linear search time for the deletion and insertion of documents.

Recently, Chen et al. (2016) propose a hierarchical clustering method based on privacy-preserving ranked keyword search method (MRSE-HCI) that clusters documents based on minimum relevance threshold and partitions next to the resulting clusters into sub-clusters, till the constraint on the maximum size of the cluster is reached. Besides, it introduces a minimum hash sub-tree structure to verify the integrity of search results. They propose a quality hierarchical clustering (QHC) algorithm based on a novel dynamic K-means which takes several rounds of calculations to obtain a stable k and that makes it somewhat inefficient. Yet, CAK-means clustering cannot overcome the instability and inefficiency problem.

Kamara and Moataz (2017) propose non-interactive highly efficient SSE schemes that handle arbitrary disjunctive and Boolean queries with worst-case sub-linear search and optimal communication complexity. The proposed construction, IEX, makes black-box use of an underlying single keyword SSE scheme that can be instantiated in various ways. To evaluate the practicality of schemes, they designed and implemented a new encrypted search framework called clusion. To resist non-volatile memory leakage attack, Chen et al. (2018) proposed a multi-keyword ranked search scheme which resists memory leakage attack (MRSS-ML), by utilizing physically unclonable functions (PUFs) to randomize the keywords and document identifiers. Owing to the noisy properties of PUFs, the fuzzy extractor (FE) is used to recover the secret keys. To further enhance the security of the proposed scheme, an order-preserving function is selected to encode the similarity scores. Wang et al. (2017) proposed a verifiable search scheme for outsourced database based on an invertible Bloom filter that is able to support efficient data update and multi-user scenarios. Fu et al. (2015) propose a tree-based index structure that supports parallel search.

However, as Poh et al. (2017) pointed out, almost all of the existing SSE schemes only concerned about the search time, though did not analyze the overhead of reading these identifiers of queried files from the disks of storage servers. It is the issue of locality in SSE schemes. Further, we believe that the issue of locality not only concerns about the read access of the files identifiers, but also concerns about the overhead of reading the files from the disks of storage servers.

Cash and Tessaro (2014) first discuss the issue of locality in SSE schemes and they define locality as the number of non-contiguous memory accesses made by the server. As plaintext search can use one contiguous access for entire postings list. They pointed out that the locality is the bottleneck of the SSE schemes and the runtime bottleneck is the disk latency but not the cryptographic processing. They also defined read overlaps as the amount of touched memory common between searches. They proved a lower bound on the trade-off between server storage size and the locality of memory accesses in the SSE, and they also gave a theoretical construction that provided a trade-off between the locality and the index size not previously achieved. Their work appears to be the first one to study the effects of security on the locality in detail.

Recently, Asharov et al. (2016) proposed three SSE schemes that follow the formalization by Cash and Tessaro (2014) and construct the first SSE schemes that simultaneously enjoy optimal locality, optimal space overhead, and nearly optimal read efficiency. Their schemes construct through a two-dimensional generalization of the classically balanced allocations ("balls and bins") problem. Demertzis and Papamanthou (2017) proposed the first searchable encryption scheme with tunable locality and linear space. Their construction can be tuned to achieve trade-offs between space, read efficiency, locality, parallelism, and communication overhead. They formally proved the security of the proposed scheme, and they also presented a thorough description of the implementation and evaluation of the scheme in external and internal memory settings. Miers and Mohassel (2017) proposed a provably secure dynamic SSE (DSSE) scheme with a significant reduction in I/O cost when used for emails or other highly dynamic materials. DSSE combines obliviously updatable index (OUI) which provides for ORAM-like properties for updates to the index with the state-of-the-art I/O-efficient SSE that indexes the full blocks. The DSSE offers a 94% savings compared to a naive implementation using ORAM.

However, the proposed MRSE-CAK scheme makes use of the inner product similarity to measure the related files which will be classified to one cluster. This is different from the SSE-1 and the SSE-2 proposed by Curtmola et al. (2006) which read one by one the related file identifiers from an array stored randomly. Thus, the locality proposed by Cash and Tessaro (2014) is different from the proposed MRSE-CAK scheme, where the operations of reading the identifiers of queried files are completed when computing the rank relevance score. Substantial importance is due to the disk latency when reading the files from the disks of storage servers. In Big Data environments, cloud servers consist of tens of thousands of machines; and load balancing and parallel read/write can improve the speed of disk I/O. At the same time, if the files that often need to be accessed in parallel are stored at the adjacent locations, equivalent to the file cache and will enormously improve the speed of file disk I/O.

Inspired by idea presented in Chen et al. (2016), we propose to utilize CAK-means clustering (Zhu et al. 2009) to improve search efficiency and accuracy. Further, since the files in one cluster are usually those files needed to be accessed at the same time, these files are stored at contiguous disk locations, which will improve the file locality and will thereby greatly improve the latency of disk I/O. It is well known that the K-means algorithm is widely used in text clustering, though the K value of K-means clustering is fixed and sensitive to its initial conditions. In order to overcome the defects of AP and K-means, the CAK-means (a combination of AP and K-means clustering) method (Zhu et al. 2009) is proposed, which makes use of affinity propagation (AP) to initialize K-means clustering. As affinity propagation (AP) algorithm identifies clusters with much lower error than other methods, the search accuracy are significantly improved. In the CAK-means based scheme, the cluster center generated by the AP algorithm is the initial clustering center of K-means; and the clustering of AP algorithm is fast and stable, which effectively improves the initial clustering center quality of K-means, overcoming the instability problem of K-means. On the one hand, the clustering method can achieve semantic search and improve the search accuracy. On the other hand, related files are stored at contiguous locations of cloud servers, and this can speed up the disk I/O and thereby improve the search efficiency of cloud servers.

3 Notations and preliminaries

3.1 Symbol definition

The symbols and notations used in this paper are as follows.

F—The plaintext file collection, denoted as a set of m files $F = \{F_1, F_2, \ldots, F_m\}$.
C—The encrypted file collection for F, denoted as $C = \{C_1, C_2, \ldots, C_m\}$.
F_v—The collection of file vectors, denoted as $F_v = \{f_1, f_2, \ldots, f_m\}$.
f_i—The i-th file vector, denoted as $f_i = \{f_{i1}, f_{i2}, \ldots, f_{in}\}$, where f_{ij} represents the weights of each keyword in file F_i.
W—The dictionary, denoted as $W = \{w_1, w_2, \ldots, w_n\}$.
C_v—The collection of cluster center vectors, denoted as $C_v = \{c_{v1}, c_{v2}, \ldots, c_{vK}\}$.
I—The clustering index which contains the relations between cluster center and the unencrypted file vectors (F_v).
Ic—The clustering index which contains the relations between cluster center and the encrypted file vectors which is attached with the identifier of the file.
F_C—The information of files classification, it includes file *id* list of a certain cluster.
Q—The query vector generated from search request.
T_Q—The encrypted form of Q, which is always called the trapdoor for the search request.
R_Q—The ranked *id* list of all files according to their relevance to query Q.
E_Q—Top-k search results.
m—The number of files in the data collection.
n—The size of dictionary W.
K—The number of clusters.
id—Unique identifier for every document.

3.2 Coordinate matching

In the proposed scheme, we utilize coordinate matching to measure the relevance between file–query, file–file, and the query and cluster centers are also considered. The relevance score between query Q and file f_i is defined by Eq. (1), the relevance score between query Q and cluster center c_{vi} is defined by Eq. (2), and the relevance score between file f_i and f_j is defined by Eq. (3).

$$S_{Qfi} = \sum_{t=1}^{n+u+1} (Q_t \times f_{i,t}) \tag{1}$$

$$S_{Qc_{vi}} = \sum_{t=1}^{n+u+1} (Q_t \times c_{vi,t}) \tag{2}$$

$$S_{f_i f_j} = \sum_{t=1}^{n+u+1} (f_{i,t} \times f_{j,t}) \tag{3}$$

3.3 Clustering based on affinity propagation K-means algorithm

A number of clustering methods have been proposed, as K-means (MacQueen 1967) and K-medoids (Huang 1998). The affinity propagation (AP) is a clustering algorithm based on the concept of "message passing" between data points (Frey and Dueck 2007), which takes as input measures of similarity between pairs of data points. Real-valued messages are exchanged between the data points until a high-quality set of exemplars and corresponding clusters gradually emerges. Unlike clustering algorithms such as K-means or K-medoids, the AP does not require the number of clusters to be determined or estimated before execution of the algorithm. Yet, similar to the K-medoids, the AP searches for "exemplars", which are the members of the input set that are representative of clusters. Such "exemplars" can be found by randomly choosing an initial subset of data points and then iteratively refining them, but this works well only if that initial choice is close to a good solution.

The K-means algorithm is widely used in text clustering. Nevertheless, the K value of K-means clustering is fixed and sensitive to its initial conditions. In order to overcome the defects of the AP and the K-means that utilize affinity propagation (AP) to initialize K-means clustering, the CAK-means

```
CAK-means Algorithm
Input: $F_v = \{f_1, f_2, \dots, f_m\}$.
Output: The clustering index $I$.
    1. Calculate similarity matrix $SM$ as follows.
        1.1 For each file, input the $i$-th file vector $f_i = \{f_{i1}, f_{i2}, \dots, f_{in}\}$
            ($1 \le i \le m$), where $f_{ij}$ represents the weights of each keyword in
            file $F_i$.
        1.2 Calculate the similarity score between document $j$ ($1 \le j \le m$) and
            document $l$ ($1 \le l \le m, j \ne l$) as Eq. (3).
        1.3 Calculate the similarity scores between every two document
            vectors and get an $m \times m$ similarity matrix $SM$.
    2. Take $SM$ as input and run AP clustering algorithm.
    3. Get the number of clusters $K$ and the initial centers for the K-
       means clustering $C_v = \{c_{v1}, c_{v2}, \dots, c_{vK}\}$.
    4. Take $C_v$ as input and run K-means clustering algorithm.
    5. Output the clustering index $I$.
```

Fig. 1 CAK-means algorithm

(combination of AP and K-means clustering) method (Zhu et al. 2009) is proposed to achieve fast searchable encryption in Big Data environments. As the AP found clusters with much lower errors than other methods, it will significantly improve the search accuracy. In the CAK-means based scheme, the cluster center generated by the AP algorithm is the initial clustering center of K-means, and the clustering of the AP algorithm is fast and stable which effectively improves the initial clustering center quality of the K-means, overcoming its instability problem. The CAK-means algorithm is illustrated in Fig. 1.

4 MRSE-CAK construction

The formal definition and concrete construction of the MRSE-CAK scheme are detailed in this section.

4.1 Definition

Definition 4.1 (*CAK-means-based multi-keyword ranked search scheme, MRSE-CAK*) A MRSE-CAK scheme consists of six polynomial-time algorithms $MRSE - CAK = (Keygen, Index, Enc, Trapdoor, Search, Dec)$, such that:

$Keygen(1^\lambda) \to (sk, k)$ it is a probabilistic key generation algorithm that is used to generate the secret key. It takes a security parameter λ as the input and returns the secret keys (sk, k).

$Index(F, sk) \to I$ it is executed by the owner to generate indexes. It takes a secret key sk and a file collection F as inputs, and returns the index I. The process of clustering is performed in this stage.

$Enc(k, F) \to C$ it is executed by the owner to encrypt the file collection using a symmetric encryption algorithm.

$Trapdoor(Q, sk) \to T_Q$ it is executed by a user to generate encrypted query vector T_Q with queried keywords and secret key.

$Search(T_Q, I_c, k_{\text{top}}) \to E_Q$ it is executed by cloud server to compute the similarity of trapdoor with an index to generate the k_{top} search results.

$Dec(E_Q, k) \to R_Q$ it is executed by user to decrypt the returned encrypted files.

4.2 Scheme construction

In the MRSE-CAK, the data owner firstly analyzes every file to fetch the keyword dictionary W. Next, all files are transformed into a collection of file vectors F_v. At the same time, file characteristics and statistic keyword frequency are extracted and a feature vector similarity matrix is constructed. After these steps, the data owner calculates the F_C and C_v using the AP K-means clustering method as presented in Sect. 3. As users request to search some keywords, the cloud server needs to query the most relevant cluster first and then use it as a benchmark to get k_{top} results. The process of file encryption, index construction, clustering, and search is illustrated in Fig. 2. Noting that the encrypted clustering index Ic contains the relations between cluster center and the encrypted file vectors which is attached with the identifier of the corresponding file.

The cloud server computes the relevance score with T_Q and index I_c, as follows.

$$\begin{aligned} T_Q \cdot I_c &= \{M_1^{-1} Q', M_2^{-1} Q''\} \cdot \{M_1^T f_i', M_2^T f_i''\} \\ &= Q' \cdot f_i' + Q'' \cdot f_i'' \\ &= S_{Qfi} \end{aligned} \tag{4}$$

After receiving the query T_Q from a user, the server computes the relevance scores using T_Q with each cluster center to get the ranked results and identify the encryption vector of cluster center c_{vi} that is closest to T_Q. The candidate set consists of c_{vi} together with its followers. According to the file similarity matrix, the cloud server can retrieve all relevant files. At this point, a threshold that is higher than the value evaluated to be relevant is set. If the results of candidate set are less than k_{top}, the cloud server continues to search for neighboring related cluster and sorts results; if the results continue to be less than k_{top}, it returns all results to the users. The complete MRSE-CAK scheme is described in Fig. 3.

In the MRSE-CAK, the secure kNN algorithm (Witten et al. 1999) is also used to encrypt file index vectors and query vectors. In order to improve the privacy of the original secure kNN algorithm, u bits of corresponding dummy keywords are added to data vectors and utilize the vector extending technique (Feingold and Varga 1962) to extend all vectors (data vector and query vector) from n dimension to $n + u + 1$ dimension.

As it can be utilized to compute the inner product similarity of an index vector I and a query vector Q, it is used

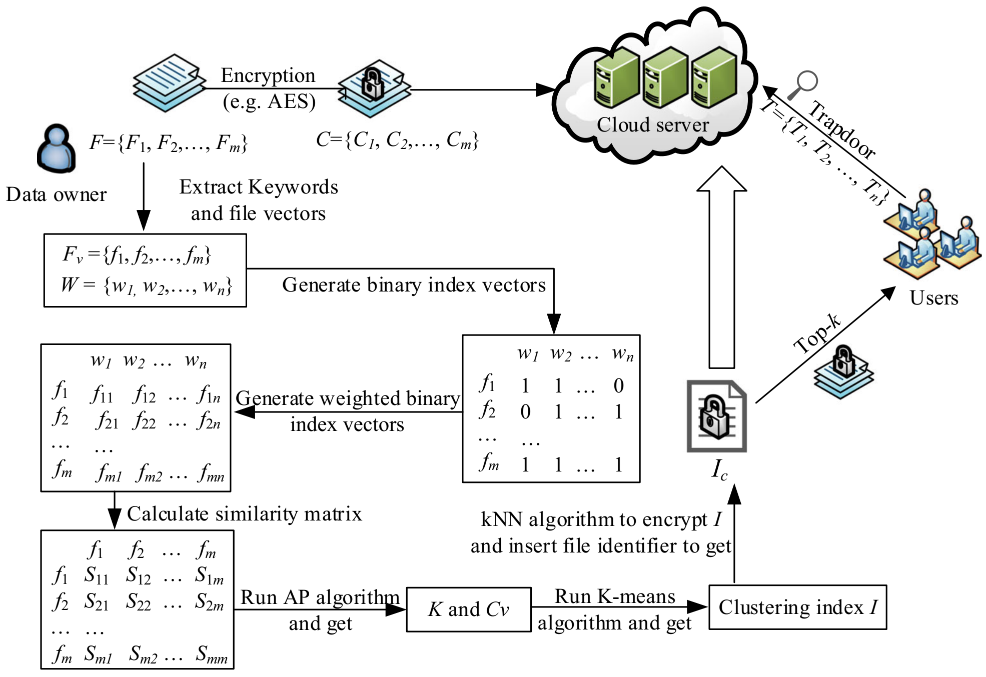
Encryption
(e.g. AES)
Cloud server
$F=\{F_1, F_2,\ldots, F_m\}$
$C=\{C_1, C_2,\ldots, C_m\}$
Data owner
Extract Keywords and file vectors
$F_v=\{f_1, f_2,\ldots, f_m\}$
$W=\{w_1, w_2,\ldots, w_n\}$
Generate binary index vectors
Generate weighted binary index vectors
Calculate similarity matrix
Run AP algorithm and get
K and Cv
Run K-means algorithm and get
Clustering index I
kNN algorithm to encrypt I and insert file identifier to get
I_c
Trapdoor
$T=\{T_1, T_2,\ldots, T_n\}$
Users
Top-k

Fig. 2 Process of file encryption, index construction, clustering, and search

to measure the relevance between the query and the file as Eq. (4). After completion of the transforming processes illustrated in Fig. 3, the data vector and the query vector cannot be recovered by analyzing their corresponding ciphertexts without any knowledge of the private key.

Note that after clustering the files collections, the data owner will encrypt the files of the same cluster and store them at the contiguous locality of disks of cloud servers. It can be achieved by issuing a continuous write operation to the disks of cloud servers.

For other issues such as data updating, existing methods such as those proposed in Xia et al. (2016) can be utilized. Due to software or hardware failure and attacks, the search results returned to users may be tampered, so the existing verifiable methods (Wang et al. 2017) can be utilized.

5 Security and performance analysis

5.1 Security analysis

The MRSE-CAK scheme is improved over the MRSE-HCI scheme, and the security analysis is presented as follows.

Theorem 5.1 *The MRSE-CAK is secure in the known ciphertext model if the MRSE-HCI scheme is secure in the known ciphertext model.*

Proof The MRSE-CAK scheme is improved based on MRSE-HCI scheme; the first difference is that the affinity propagation (AP) is considered to initialize K-means clustering which improves the initial clustering center quality of K-means effectively. As affinity propagation found clusters with lower error than other methods, it will significantly improve the search accuracy. The second difference is that the related files are stored at contiguous locations of cloud servers, in which does not weaken the security. When a user queries some keywords, the server knows which encrypted files do satisfy this query, no matter whether the files are dispersely stored on the server or continuously stored on the server.

To protect the data privacy, secured symmetric encryption techniques are utilized to encrypt all plaintext files. Nevertheless, discussions on the security of these techniques are not the scope of this paper. We assume that the secret key *sk* is randomly generated, and if the secret key *sk* is kept confidential, the index privacy can be well protected. The vector encryption method has been proved to be secure in the known ciphertext model (Chen et al. 2016). The MRSE-CAK scheme utilizes the same vector encryption as MRSE-HCI. Since only the cluster methods are improved, no more information will be leaked to the curious-but-honest cloud server than MRSE-HCI.

As MRSE-CAK is a top-k ranked search scheme, the rank order of the search results is revealed. To protect this privacy, some effective methods, such as private information retrieval (PIR) technique (Ishai et al. 2006) can be adopted. Moreover, as cloud servers are in charge of most computations, secure hardware can be utilized to protect the privacy. □

```
Keygen($1^\lambda$):
    1. The data owner randomly generates an $(n+u+1)$ bit vector $S$ and two invertible $(n+u+1) \times (n+u+1)$ dimension matrices $\{M_1, M_2\}$, $sk = \{S, M_1, M_2\}$.
    2. Randomly generates the $n$-bit secret key $k$.
Index($F$, $sk$):
    1. Input the secret key $sk$ and data set $F$.
    2. Extract a dictionary $W$ from $F$.
    3. For each file $F_i$ in $F$, it constructs a file vector $f_i$.
    4. Invoking CAK-means algorithm to get a plaintext clustering index $I$.
    5. Extend the dimension of $f_i$ from $n$ to $(n+u+1)$ by splitting it into two vectors as follows:
    if the $j$-th of $S$ is 0,
            $f_i'[j] = f_i''[j] = f_i[j]$;
        else
            $f_i''[j]$ is set to a random number and set $f_i'[j] = f_i[j] - f_i''[j]$.
    6. Encrypt index as $\{M_1^T f_i', M_2^T f_i''\}$, and insert corresponding file identifier and get $I_c$.
Enc($k$, $F$): It is executed by data owner to encrypt the file collection using any secure symmetric
    encryption mechanism (the AES is used in this scheme) to encrypt the plaintext files to get
    ciphertexts. Note that the data owner will encrypt the files of the same cluster and store them at
    the contiguous locality of disks of cloud servers.
Trapdoor($Q$, $sk$):
    1. Data user sends the searched keywords to the data owner. After analyzing the query, the data owner will build the query vector $Q$ with the dictionary $W$.
    2. Extend the dimension of $Q$ from $n$ to $(n+u+1)$ by splitting it into two vectors as follows:
    if the $j$-th of $S$ is 1,
            $Q'[j] = Q''[j] = Q[j]$;
        else
            $Q'[j]$ is set to a random number and set $Q''[j] = Q[j] - Q'[j]$.
    3. Finally, the trapdoor is generated as $T_Q = \{M_1^{-1}Q', M_2^{-1}Q''\}$ and it is sent back to the data user.
Search($T_Q$, $I_c$, $k_{top}$):
    1. After receiving the query $T_Q$ from data user, the cloud server computes the relevance score with $T_Q$ and index $I_c$ as Eq. 4.
    2. The server chooses the most matched cluster which has the highest relevance score.
    3. For each file contained in the matched cluster, the cloud server extract the corresponding encrypted file vector from index $I_c$. Then it returns the $k_{top}$ files according to the file similarity.
Dec($E_w$, $k$): After receiving the $k_{top}$ encrypted files, the data user decrypt the ciphertexts $E_Q$ to get
        plaintext files using the secret key $k$.
```

Fig. 3 The MRSE-CAK scheme

5.2 Performance analysis

In this section, experiments to evaluate the proposed MRSE-CAK scheme are designed and performance results compared next to MRSE-HCI (Chen et al. 2016).

5.2.1 Comparison between sequential and random I/O

As the MRSE-CAK scheme utilizes CAK-means to cluster all related files to one cluster, these files are stored at contiguous locations of cloud servers to improve the file locality. We will compare the I/O performance of sequential and random disk read/write operations.

IOR is used to issue sequential and random requests on a 100 GB SSD (OCZ-REVODRIVE X2) and a 250 GB HDD (SEAGATE ST32502NSSUN250G). To make the evaluation in this research fair and conservative, the operating system buffer caches are flushed before each execution to ensure that all data will be read from the storage devices. Moreover, the "dirty" data in the memory are flushed to the storage devices to ensure that write throughput on the storage devices are correctly measured. Figure 4 shows the results under different request sizes from 1 to 256 KB. As depicted in Fig. 4, the sequential I/O performance is much better than that of random request for both read and write operations for both HDD and SSD.

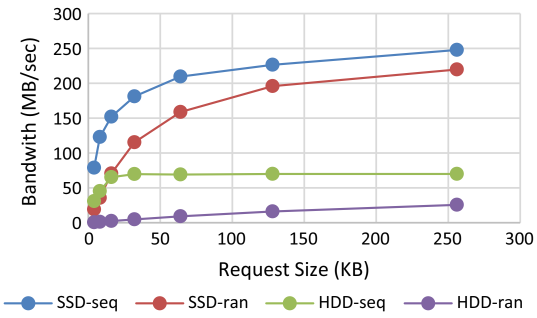

Fig. 4 Comparison between sequential and random I/O

The following experiments are implemented in MATLAB on a computer server configured with Intel core i5-7200U

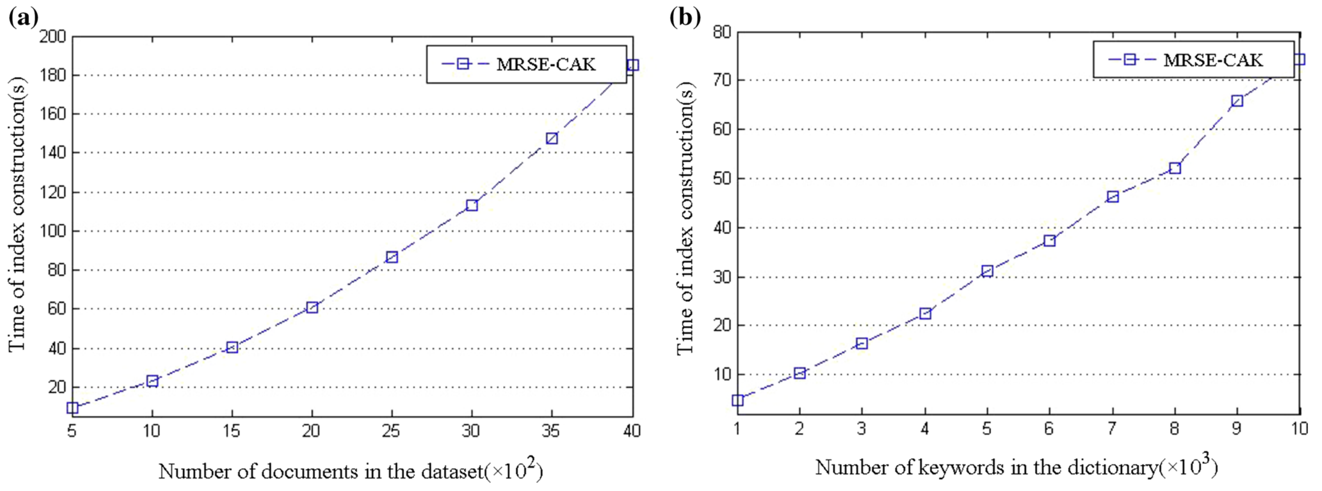

Fig. 5 Time overhead of index construction. **a** For different size of data set with the same dictionary, $n = 4000$. **b** For same data set with different size of dictionary, $m = 1000$

2.5 GHz processor, 8 GB memory, AMD Radeon R7 M445 graphics card, and Win10 (64 bit) operation system. Gauss matrix, invertible matrix, and matrix multiplication are achieved by using MATLAB libraries. As defined in Sect. 3, n denotes the dictionary size, m denotes the number of documents in the data set, and q denotes the number of keywords users requested. Since the AP algorithm and the K-means algorithm have been integrated in MATLAB libraries, the proposed implementation needs to invoke them as well to modify parameters.

5.2.2 Overhead of index construction

The process of index construction is achieved in three main steps, as step (1): mapping the keyword dictionary extracted from each file to a binary vector, step (2): invoking CAK-means algorithm to build index, and step (3): encrypting the index by splitting and multiplying of two matrices. The dimension of vectors depends on the size of the dictionary and it directly determines the time of mapping encryption. The time of generating complete index is related to the number of files in data set F and the number of keywords in dictionary W. Figure 5a shows that given the same dictionary ($n = 4000$), the time overhead of index construction increases linearly with the increase in dataset, and Fig. 5b demonstrates that given the same number of files ($m = 1000$), the time overhead is determined by the size of keyword dictionary for index construction. Thus, as the process of index construction of MRSE-CAK is similar to MRSE-HCI, the time overhead is similar.

5.2.3 Trapdoor generation

Similar to index construction, the generation of each trapdoor comprises the process of vector splitting and the multiplication of two matrices. These operations can be implemented by MATLAB. In comparison to index construction, trapdoor generation consumes relatively less time. Figure 6a shows that the time of generating a trapdoor is greatly affected by the number of keywords in the dictionary. Moreover, the number of query keywords has minimum influence on the overhead of trapdoor generation when the dictionary size is fixed, as shown in Fig. 6b.

5.2.4 Search efficiency and accuracy

The search efficiency and accuracy of the proposed MRSE-CAK scheme with MRSE-HCI are compared and evaluated with different conditions, as depicted in Fig. 7. Figure 7a shows that the search time of MRSE-CAK with respect to the size of data set is comparable with MRSE-HCI. It is noteworthy that the proposed MRSE-CAK scheme has a higher efficiency than MRSE-HCI. In addition, it is shown in Fig. 7b that the search time of the MRSE-HCI scheme keeps stable with the increasing query keywords. Meanwhile, the search time of the proposed MRSE-CAK scheme is lower than MRSE-HCI.

Same evaluation method is considered to evaluate the search accuracy as MRSE-HCI, in which the search results are based on a relevance threshold set in advance. Only the files whose relevance score is higher than that threshold is added to the set of search results. The search process is illustrated as Fig. 3 and return k_{top} search results. The relevance of different retrieved files is evaluated using Eq. (5). At this point, r denotes the number of files retrieved by plaintext search, Score() is a function to compute the relevance score.

$$Pd = \sum_{j=1}^{k_{\text{top}}} \sum_{i=1}^{k_{\text{top}}} \text{Score}(f_j, f_i) \Big/ \left(\sum_{j=1}^{r} \sum_{i=1}^{r} \text{Score}(f_j, f_i) \right) \quad (5)$$

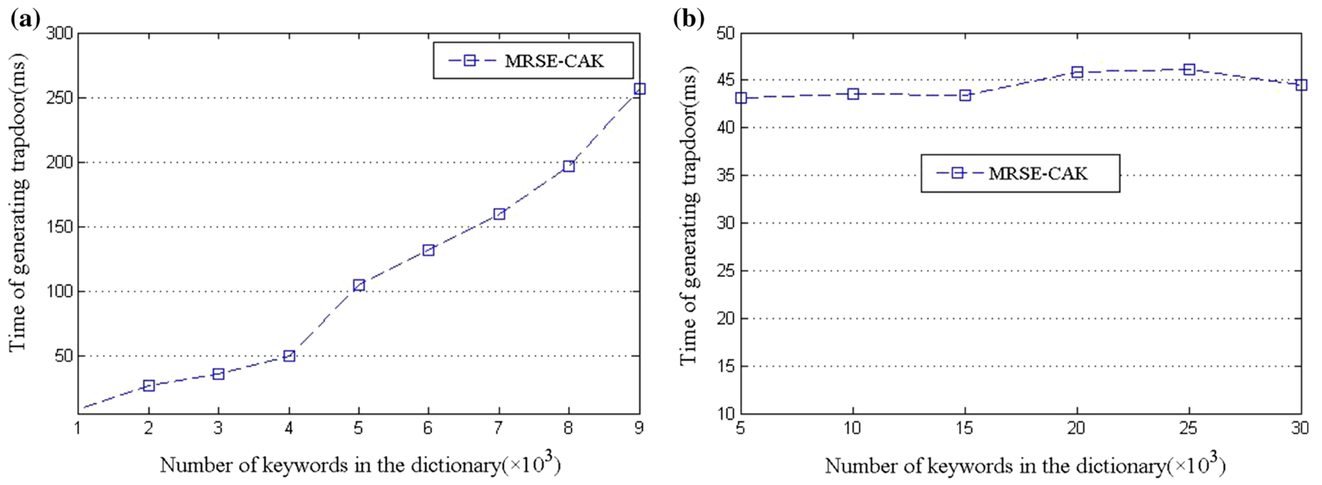

Fig. 6 Time overhead of trapdoor generation. **a** For different sizes of dictionary with the same query keywords, $q = 20$. b For different numbers of query keywords with the same dictionary, $n = 4000$

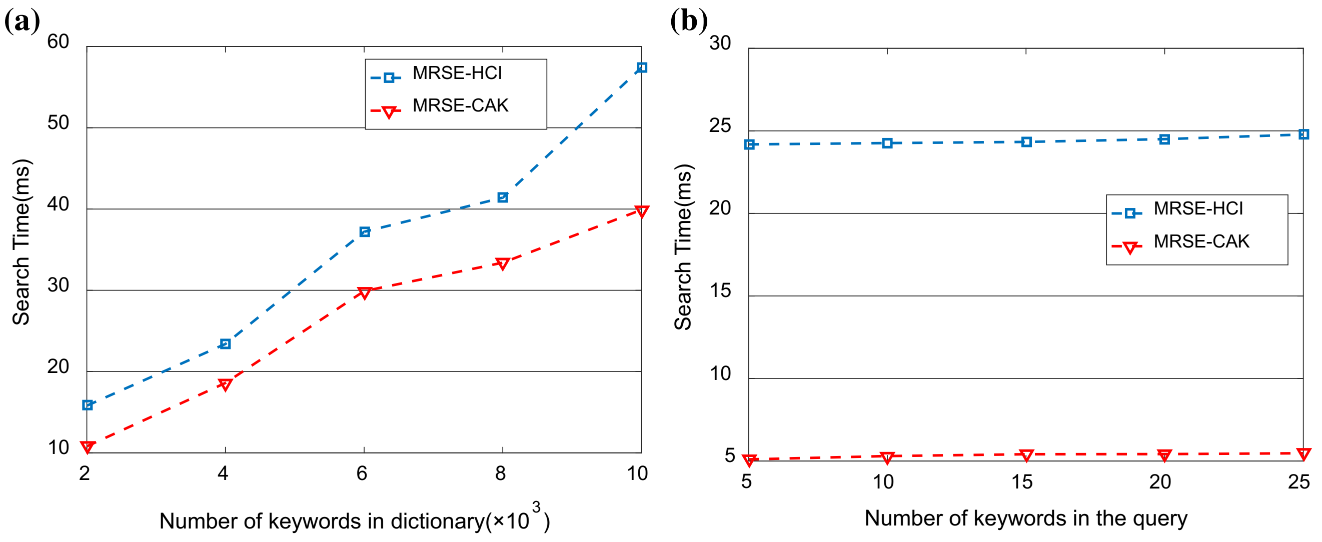

Fig. 7 Search time. **a** For the same query keywords in different sizes of dataset, $q = 20$, $n = 2500$. **b** For different numbers of query keywords in the same dataset, $m = 1000$, $n = 2500$

The relevance of retrieved files and the query is evaluated using Eq. (6).

$$Pq = \sum_{i=1}^{k_{\text{top}}} \text{Score}(TQ, fi) \Big/ \sum_{i=1}^{r} \text{Score}(Q, fi) \tag{6}$$

The search accuracy is depicted in Fig. 8, where the relevance of retrieved files with the size of the file set increasing from 2000 to 10,000 is also illustrated in Fig. 8a. We can observe that the relevance between files in MRSE-CAK is slightly higher than that of MRSE-HCI, signifying that the retrieved files generated by MRSE-CAK are much closer to each other. Similarly, Fig. 8b shows that the relevance between the query and retrieved files of MRSE-CAK is better than that of MRSE-HCI.

Likewise in MRSE-HCI with a sharp increase in files in the data set, the search time of MRSE-CAK scheme increases linearly, whereas the traditional method increases exponentially. Moreover, it also has the advantage over the traditional method in the rank privacy and relevance of retrieved files. As overall, the results of the experiments show that the proposed MRSE-CAK scheme improves the search efficiency and accuracy while ensuring equivalent security.

6 Conclusions and future work

In this paper, we propose to utilize fast clustering of the AP algorithm to improve file locality and realize an efficient multi-keyword ranked searchable symmetric encryp-

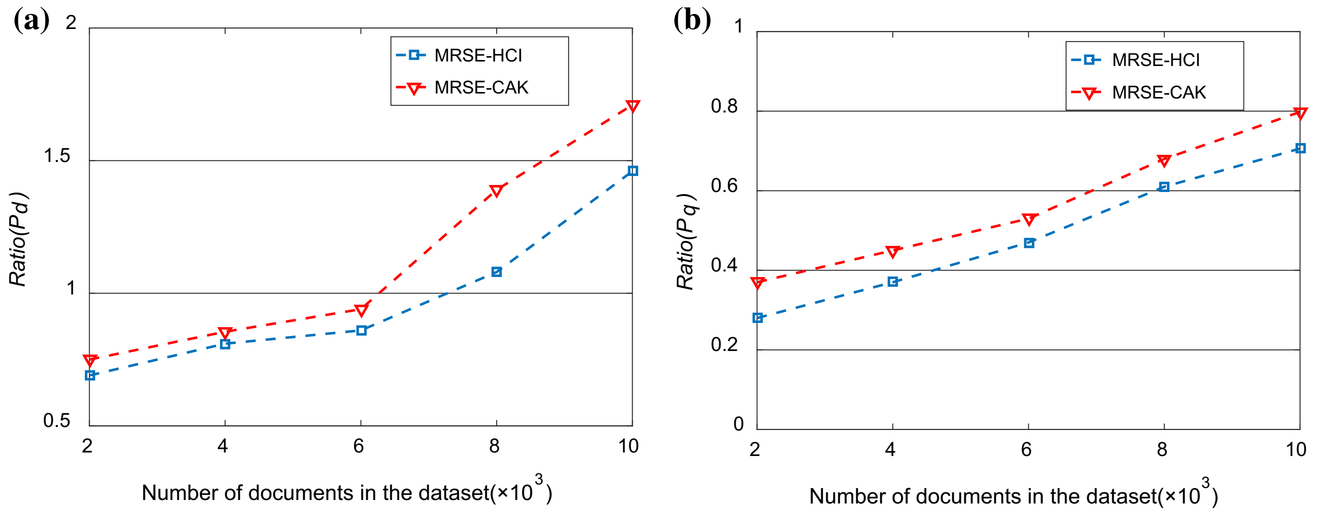

Fig. 8 Search accuracy. **a** Relevance of documents. **b** Relevance between documents and query

tion scheme MRSE-CAK that can take advantages of the relationship among files to speed up the search process. The CAK-means cluster method is introduced to enhance the performance of semantic search and overcome the instability problem of K-means. In MRSE-CAK, datasets are divided into several clusters, and the related files in one cluster are stored at the contiguous locality of disks that will largely improve the file locality and concurrently speed up the read and write of disk I/O. Coordinate matching is utilized to evaluate the similarity between outsourced files to achieve accurate ranked search. Considering privacy preserving, the MRSE-CAK scheme is secure in the known ciphertext threat model. From experiment results, we show that the proposed MRSE-CAK scheme improves the search efficiency and accuracy while ensuring the equivalent security.

Future relevant studies of this research include detection of bottleneck, which could be the latency on reading the index or the latency on reading files from the disk. In addition, improvements on the efficiency of the search algorithm and the design of a secure scheme in enhanced threat model will also be aimed.

Acknowledgements This work was supported by the Natural Science Foundation of China (Nos. 61602118, 61572010 and 61472074), Fujian Normal University Innovative Research Team (No. IRTL1207), Natural Science Foundation of Fujian Province (Nos. 2015J01240, 2017J01738), Science and Technology Projects of Educational Office of Fujian Province (No. JK2014009), and Fuzhou Science and Technology Plan Project (No. 2014-G-80).

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Asharov G, Naor M, Segev G, et al (2016) Searchable symmetric encryption: optimal locality in linear space via two-dimensional balanced allocations. In: Proceedings of the international conference on ACM symposium on theory of computing, Cambridge, MA, USA, pp 1101–1114

Cao N, Wang C, Li M et al (2014) Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans Parallel Distrib Syst 25(1):222–233

Cash D, Tessaro S (2014) The locality of searchable symmetric encryption. In: Proceedings of the international conference on the theory and applications of cryptographic techniques, Copenhagen, Denmark, pp 351–368

Chen C, Zhu X, Shen P et al (2016) An efficient privacy-preserving ranked keyword search method. IEEE Trans Parallel Distrib Syst 27(4):951–963

Chen L, Qiu L, Li KC et al (2017) DMRS: an efficient dynamic multi-keyword ranked search over encrypted cloud data. Soft Comput 21(16):4829–4841

Chen L, Qiu L, Li K-C, Zhou S (2018) A secure multi-keyword ranked search over encrypted cloud data against memory leakage attack. J Internet Technol 19(1):179–188

Curtmola R, Garay J, Kamara S, et al (2006) Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the international conference on ACM conference on computer and communications security, Alexandria, VA, USA, pp 79–88

Demertzis I, Papamanthou C (2017) Fast searchable encryption with tunable locality. In: Proceedings of the international conference ACM international conference on management of data, Chicago, Illinois, USA, pp 1053–1067

Feingold DG, Varga RS (1962) Block diagonally dominant matrices and generalizations of the Gerschgorin circle theorem. Pac J Math 12(4):1241–1250

Frey BJ, Dueck D (2007) Clustering by passing messages between data points. Science 315(5814):972–976

Fu Z, Sun X, Liu Q, Zhou L, Shu J (2015) Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans Commun 98(1):190–200

Huang Z (1998) Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min Knowl Disc 2(3):283–304

Ishai Y, Kushilevitz E, Ostrovsky R (2006) Cryptography from anonymity. In: Proceedings of the international conference on foundations of computer science, Washington, DC, USA, pp 239–248

Kamara S, Moataz T (2017) Boolean searchable symmetric encryption with worst-case sub-linear complexity. In: Proceedings of the international conference on the theory and applications of cryptographic techniques, Paris, France, pp 94–124

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the international conference on Berkeley symposium on mathematical statistics and probability, California, USA, pp 281–297

Miers I, Mohassel P (2017) IO-DSSE: scaling dynamic searchable encryption to millions of indexes by improving locality. In: Proceedings of the international conference on network and distributed system security symposium, San Diego, California, pp 1–13

Poh GS, Chin JJ, Yau WC et al (2017) Searchable symmetric encryption: designs and challenges. ACM Comput Surv 50(3):40

Wang J, Chen X, Li J et al (2017) Towards achieving flexible and verifiable search for outsourced database in cloud computing. Future Gener Comput Syst 67:266–275

Wang B, Yu S, Lou W, et al (2014) Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In: Proceedings of the international conference on computer communications, Toronto, Canada, pp 2112–2120

Witten IH, Moffat A, Bell TC (1999) Managing gigabytes: compressing and indexing documents and images. Morgan Kaufmann Publishing, San Francisco

Xia Z, Wang X, Sun X et al (2016) A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans Parallel Distrib Syst 27(2):340–352

Zhu Y, Yu J, Jia C (2009) Initializing K-means clustering using affinity propagation. In: Proceedings of the international conference on hybrid intelligent systems, Shenyang, China, pp 338–343