Multi-owner Encrypted Ranked Keyword Search Using Machine Learning Techniques

Laila Tul Badar[1], Chungen Xu[1](✉), and Ali Zakir[2]

[1] School of Science, Nanjing University of Science and Technology, Nanjing, China
{lailatulbadar,xuchung}@njust.edu.cn
[2] School of Computer Science and Engineering, NJUST, Nanjing, China
alizakir@njust.edu.cn

Abstract. In the present era, the prevalence of network technology and cloud computing capabilities for multiple owner model has drawn much attention. Data owners outsource their data to the cloud and enjoy convenient services. However, extending single owner model into multiple owner model still have some issues, such as dimension reduction issue, high communication overhead, and efficient search in Searchable Symmetric Encryption (SSE) remain a challenging task. Integration of machine learning methods with the framework of searchable Encryption (SE) is a diverse way to solve these problems. In this paper, we developed Multi-owner encrypted ranked keyword search using Machine learning techniques(MERS-ML). Our Scheme utilized principal component Analysis (PCA) model to reduce high-dimensional data into low-dimensional codes and enabled low-overhead system maintenance. K-means clustering approach is used to solve the problem of the quality of different document of multiple owner model. To achieve fast query and efficient search relatively close to *O(log N),* we designed search balanced index tree. Besides, attribute-based encryption is used to achieve convenient key management as well as authorized access control. Compared with previous work, our scheme provide adequate privacy protection, improved search efficiency, and introduce low overhead on computation and storage.

Keywords: Searchable encryption · Ranked keyword search · Multi owner model · Binary index tree · Principle component analysis

1 Introduction

Background. As a promising computing paragon, cloud computing has become a hot topic for research and industrial communities. It brings huge benefits to data owner such as, provide inclusiveness, flexibility, scalability, and rapid retrieval of data. Due to its highly desirable features, organizations, as well as individuals, are influenced to outsource their data onto the cloud server to accomplish ease and low management cost. Unfortunately, cloud computing is facing many problems and challenges during the transaction and storage of the outsourced data. Outsourced data onto the cloud contain sensitive information such as medical records, organization's financial records etc. Illegitimate use of client personal information or reveal any sort of private data is seemingly occur as data is stored in third-party cloud server. For the safety, security, and privacy of data, researchers have been paid much attention to the growing security incidents in cloud computing. To solve these issues, data must be encrypted before outsourced into the cloud server.

Related work and Challenges. Song et al. [1] proposed the Searchable Symmetric Encryption (SSE) model that supported effective keyword search over encrypted data along with the assurance of security and privacy of data. Curtmola et al. [2] provided security against adaptive and non-adaptive chosen-keyword attacks. Boneh et al. [3] proposed public-key encryption system with a keyword search. Xia et al. [4], proposed a balanced binary tree to build the index by using Greedy Depth First Search (GDFS) algorithm. Sun et al. [5] proposed TF * IDF keyword weight generation algorithm for better search accuracy, single keyword ranked search [6], fuzzy keyword search [7, 8], Dynamic keyword search [4], and Multi keyword ranked search [9] are the different flavor of public-key encryption that have been published with better security efficiency and query experiences. To achieve sublinear search time, Cao et al. [9] first proposed Multi keyword Ranked Search over Encrypted cloud data (MRSE) for single data owners and establish strict privacy requirements. However, due to high computation of matrix operations the performance of the model is decreased. Ranked keyword search scheme enhanced system usability by returning the most relevant documents in ranking order. So, users only need to decrypt a small number of ciphertext to achieve the desired documents. Though, these schemes only support searchable encryption for the "Single Owner" model. Owing to the diverse demand of application scenarios, users wanted to share their data on a cloud having multiple characteristics and host a large number of documents outsourced by Multiple Owners Model [10–13]. Multiple owner model is still facing some unsettled issues related to secure keys. It is difficult to design secure keyword search for multiple owner model because it uses their self-chosen secret keys and not share it with any other data owner which brings some major unresolved problem. These problems are, **(1)** Dimension disaster in the system overhead during index construction, trapdoor generation, and query execution is mostly determined by matrix multiplication. In other words large number of data from different data owners because high dimensionality issues leads to a results that searchable encryption cannot put into use in real world scenarios. **(2)** Data users need to generate multiple trapdoors for multiple data owners even for the same query. That's why SSE is still incompetent to satisfy user requirements. To solve these shortcomings, we need to examine the roots of the problems and find best conceivable solutions. Machine learning is so far the current popular technique that only supports the plain texts and didn't support the encrypted datasets. So, it is necessary to discuss both the problems and come up with high efficiency into SE model. Gou et al. [10] define multi-keyword ranked search scheme for multiple data owner model (MRSE-MO), and then design a keyword document and owners (KDO) weight generation algorithm. Experimental results showed that (KDO) weight generation algorithm is better than the traditional TF * IDF algorithm. When user wants to search for multiple owner model, the quality of different document from different owners are not same even if they are about the same area. In that situation, the operation of calculating similarity face dimensionality problem which bound the accessibility of the system. Last but foremost, it ignores the security and privacy in known Background Model (threat model, try to reveal the private data in SE system) [9].

In this paper, we focused on principle component analysis (PCA) [14] one of the lightweight unsupervised machine learning algorithm to solve the curse of dimensionality issue. It has been widely used in different fields of digital signal processing, robotic sensor data, and image processing. PCA applies a linear transformation to the data that allow variance within the data to be expressed in term of orthogonal eigenvectors. We used K-means clustering [15] algorithm, to solve the problem of quality of different documents in multiple owner model. To obtain high search efficiency, we proposed a balanced binary index tree and the Greedy Depth-first search (GFDS) algorithm generated by probabilistic learning. We performed random searches and get the sum of the relevant score of each index and after that sort it according to the score from high to low. Secure k-nearest neighbor (KNN) [16] algorithm is constructed to encrypt query and index vectors and ensure the security between the index and query vectors.

1.1 Our Contributions

The major contribution of our system are manifold:

- PCA algorithm is used to reduce the dimension of query and index vector in Multi Owner Model to enhance user experience and reduce system overhead.
- We construct binary index tree structure that followed the bottom-up Greedy-Depth-First-Search (GDFS) approach, and sorted the node by maximum probability. The complexity of the binary index tree are closed to *O(log N),* proved that the query speed is faster and more stable than the previous work [9, 10].
- We proposed a K-means clustering approach, to solve the problem of document quality differences between different data owners.
- The extensive experiments on real-world dataset further shows that our scheme is indeed efficient and achieve effectiveness.

1.2 Organization

The rest of the paper is organized as follows. In Sect. 2, some basic notations, system model, threat models, and design goal of scheme is introduced. Brief construction of scheme is given in Sect. 3. Experiments and performance analysis is given in Sect. 4 and 5. In Sect. 6 we describe the conclusion of this paper.

2 Notation of Problem Formulation

Table 1. Introduced some important notations for understanding formulation and statement in this paper.

Table 1. Notations and Description

Notations	Description
DO	Represent Data Owners. It consists of n number of Data Owners such as $DO = (DO_1, DO_2 \ldots\ldots DO_n)$.
F	The collection of plaintext documents owned by DO_i is denoted as $F = (F1, F2 \ldots Fn)$ and the sequence of keywords is considered as $F_{i,j}$
C	The encrypted document collection F_i that store in the cloud server and expressed as $C =$ (C1,C2 . . . Cn).
D	The dictionary specifically the set of keywords denoted as $(d_1,\ d_2 \ldots d_m)$ where D is used as a public list shared by all participants in our scheme. And M is the size of the dictionary
P_i	The *n-length* popularity vector associated with C.
I	The collection of index vector set of documents $F_{i,j}$where t^{th}a bit of I means that $I[t] = 1$ the document has the keyword else $I[t]\ = 0$.
$\tilde{I}_{i,j}$	The update weighted index vector of documents after standardized processing
τ	Plaintext form of a balanced binary tree for the document of all data owners.
$\tilde{\tau}$	The encrypted form of a balanced binary tree for the document of all data owners.
Q	Weight query vector generated based on query request and represented as $Q = (Q_1 \ldots\ldots Q_m)$.
TD	The trapdoor which is the encrypted form of Q and calculated by data users.

2.1 Proposed Framework

There are three major entities involved in our proposed framework which are Data Owners *(DO),* Data Users *(DU),* and Cloud Server *(CS)* as illustrated in Fig. 1.

- *DO* has aspired to outsource the collection of documents *D* to a cloud server. But before that they need to encrypt the documents *D* to the encrypted form of *C*, and outsourced encrypted searchable index *C* to the cloud server.
- *DU* wants to access the files in which he/she are interested in the cloud data provided by all *DO. DU* generate the trapdoor and number *k* to search over encrypted data on cloud server, and retrieve *top-k* relevant documents. The retrieved Encrypted documents are decrypted with the secret keys.

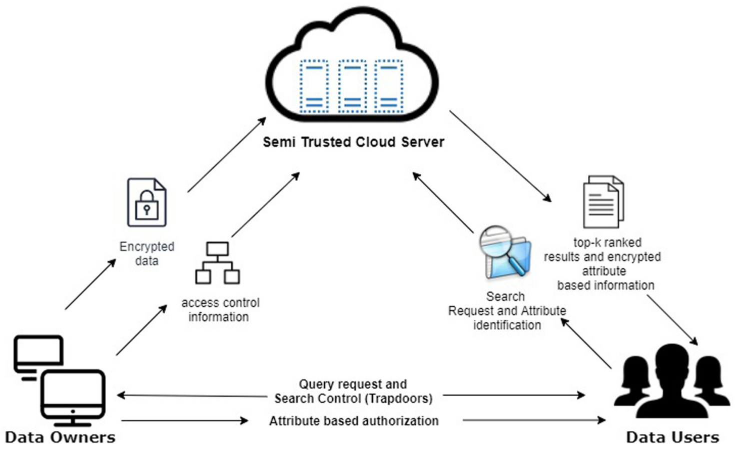
Semi Trusted Cloud Server
Encrypted data
access control information
top-k ranked results and encrypted attribute based information
Search Request and Attribute identification
Query request and Search Control (Trapdoors)
Attribute based authorization
Data Owners
Data Users

Fig. 1. The basic architecture of (MERS_ML)

– *CS* store the encrypted documents *C* and an encrypted searchable index for data owners, Based on request query from *DU*, it provide encrypted data with the most relevant *top-k* documents.

2.2 Threat Model

A cloud server is considered "honest-but-Curious" in SE, it is semi-trusted threat entity [13] Specifically, *CS* works honestly and correctly and executes the commands in the delegated protocol. However it is curious to infer and analyze received data (including index), and try to identify private information of encrypted data and carry out an attack. Depending on what information the cloud server knows, there are two threat models, which are discussed below;

– *Known ciphertext model.* In this model, the *CS* only knows the encrypted document collection, the searchable index outsourced by *DO*, trapdoors, and number *k* outsourced by *DU*. *CS* conducts ciphertext only attack (COA) [17] and try to destroy the privacy of *DO* and *DU*.
– *Known background model.* This model is stronger than Known ciphertext model. *CS* knows more statistical information about dataset such as encrypted documents size, encrypted index, trapdoors and their corresponding search results. *CS* try to learn the location of newly added entries as they are stored in lexicographical order in indexes.

2.3 Design Goals and Security Definitions

To ensure the correctness completeness and efficiency of ranked multi keyword scheme over encrypted cloud data outsourced by *DO* our system aim to achieve the following security goals:

1. *Ranked Multi keyword retrieval for multiple owners.* The proposed scheme not only support multiple owner model (which are encrypted with multiple keys for multiple data owners) but also support ranked keyword search. The design scheme retrieved all the matching documents and return *top-k* results to *DU*.
2. *Search efficiency.* The search efficiency improved by constructing the binary index tree and achieved the query complexity closer to *O(log N)* [18]
3. *Security.* It provided Security under the threat models, as we discussed above, the scheme prevent the semi-trusted *CS* from learning additional information and fulfill the following security requirements
   - *Index privacy.* The documents and encrypted keywords indexes of any data owner are protected from *CS*
   - *Query privacy.* The *CS* did not collect and identify the plaintext form of keyword information through the trapdoor
   - *Trapdoor Unlinkability.* The trapdoor unlinkability required that the trapdoor generation algorithm are randomize instead of deterministic and *CS* recognized whether two trapdoor quires generated form the same search request.

3 The Design of MERS-ML

In this section, we describe (MERS-ML) framework by using secure (KNN) [16] model, consist of 5 probabilistic-polynomial-time algorithms (PPT) are discussed below,

$\boldsymbol{K \leftarrow keyGen}$**:** It is a probabilistic index generation algorithm run by data owners. *DO* generate the secret key (*SK*), where $SK' = \{S', M_1', M_2'\}$.M_1 and M_2 are two invertible matrices their dimensions are $(n \otimes n)$ and S is a random $n - length$ vector.
$\boldsymbol{Updated - KeyGen(SK', \ell)}$**:** For updating the keyword list and adding the new keyword ℓ into the dictionary D. DO_i generate new. $SK' = \{S', M_1', M_2'\}$.M_1' and M_2' are two invertible matrices their dimensions are $(n + \ell) \otimes (n + \ell)$ and $S\prime$ are a random $(n + \ell) -$ *length* vector. This support dynamic operations into the scheme.
$\boldsymbol{I \leftarrow Buildindex(F, SK)}$**:** *DO* generate the searchable indexes for documents and add random noise into the weighted index vector I to obtain security under known background model [14]. The weighted index I and SK build an encrypted index tree $\widetilde{\tau}$ based on the dataset F. Finally DO_i send the encrypted index tree $\widetilde{\tau}$ to *CS*.
$\boldsymbol{TD \leftarrow GenTrapdoor(Q, SK)}$**:** *DU* send query request to *DO*. *DO* generate a query $Q = \{Q_1 \ldots . Q_n\}$ build the *TD* using *SK* and send Trapdoor *TD* to *CS*.
$\boldsymbol{Q \leftarrow Query(TD, K, I)}$**:** *DO* send the trapdoor information and query instruction to *CS*. When the *CS* received query request, it perform ranked search on the index I with the help of *TD* and finally return *top-k* documents to *DU*.

3.1 Secure and Efficient MERS-ML Details

Plaintext form of binary index vector generation. According to the Keywords of documents and dictionary D, the *DO* build the index vector I of the binary form for his/her document. Then send the index vector to *CS*. It is also the classical expression of the vector space model (VSM) [19].

Index dimension reduction. Dimension disaster exacerbates redundancy in document vector, and cause computational burden on the system. As the number of data increases in multi-owner model, the data features become very sparse, which made keyword dictionary for index construction very large and leads to dimensionality issues in computing for all index vector form different owners. We used PCA [14] to overcome these issues which improve low query efficiency and increase the search efficiency of the scheme.

(1) We have a dataset X, $i = \{1, 2 \ldots ..m\}$, in the first step we need to preprocess the data. We normalized the data by calculating the mean $\bar{x}$ and subtracting the mean $\bar{x}$ from each of the data point x_i

$$xi - \bar{x} \tag{1}$$

(2) Calculated the covariance matrix that is symbolized with C.

$$Cij = \frac{\sum_{i=1}^{n} (xi - \bar{x})\,(xi - \bar{x})}{(n-1)} \tag{2}$$

(3) Determine the eigenvalues and the corresponding eigenvectors of the covariance matrix to identify the principal components. C is the symmetric matrix so a positive real number λ and a non-zero vector v can be found such that

$$Cv = \lambda v \tag{3}$$

Where λ is a scalar value called the eigenvalue, and v is the eigenvector of C. To find non-zero v apply the singular value decomposition (SVD) method to solve the equation $|C - \lambda I| = 0$. If C is $m \times m$ matrix of full rank, n eigenvalues can be found $\lambda 1, \lambda 2 \ldots \lambda n$, and using $|C - \lambda I|v = 0$ all the corresponding eigenvectors found.

(4) After the eigenvectors obtained, we ranked it in decreasing order of eigenvalues, small eigenvalues mean that there components are less important, So we ignored them without losing important information and choose the first k (number of components).
Eigenvectors yielded the new k dimensions. Finally, obtain a new feature vector consisting of the eigenvector of principle component.

(5) In a final step, we need to modify our samples by re-orientating data from original one to the ones representing by principle components.

$$ReduceData = FeatureVector \times ScaledData \tag{4}$$

Here *FeatureVector* is matrix with the eigenvectors in the column transposed, so the eigenvectors now in rows. *Scaled Data* is the mean-adjusted data transposed and *Reduce Data* is the final dataset. After dimension reduction, we clustered the documents by using k-means approach and perform the clustering on all *DO* index vectors. We divided the keyword dictionary into multiple sub-dictionaries and in this sense large number of high similarity index vectors found, which solve the problem of document quality between different data owners. After getting the final data the length of index vectors become shorter than before and *DO* obtain a new binary index $\hat{I} = \{\hat{I}1 \ldots\ldots \hat{I}s\}$ with lower dimension size.

Secure Weight index generation. (1) *Correlativity matrix generation.* To construct the secure weight index for new binary index vector $\hat{I}$, and calculate the keyword weight precisely, it is compulsory to consider the semantic relationship between keywords that access the degree of influence among different keywords. We used the corpus to find the relevance between different keywords. We represented relevance between the keywords by obtaining the correlativity matrix $S_{M \times M}$ (symmetric matrix). **(2)** *Weight generation.* After obtaining the Correlativity matrix S, we use KDO [15] weight generation algorithm designed for constructing the weight of different data owners about different keywords and design average keyword popularity (AKP) about different DO. The AKP for single owner DO_i is computed as $AKP = (P_i \cdot I_i) \otimes \alpha_i$. Where $\alpha_i(\alpha_{i,1}, \alpha_{i,2} \ldots \alpha_{i,n})$ is an n-length (n is the size of dictionary),Pi is the n-length vectors (where n represent the number of document in F), I_i indicate the index vector of document $F_{i,j}$ and the operator $\otimes$ represent the product of two vectors corresponding elements, as if $L_i(d_t)| \neq 0$ then $\alpha_{i,t} = \frac{1}{|Li(dt)|}$ else $\alpha_{i,t} = 0$. Where $L_i(d_t)|$ is the number of document containing d_t. Based on correlation assumptions calculate the raw weight information for data owner DO_i is denoted as $W_i^{raw} = S \cdot AKP_i$ where $W_i^{raw} = (w_{i,1}^{raw}, w_{i,2}^{raw} \ldots\ldots w_{i,m}^{raw})$ **(3)** *Normalized weight.* All the keywords in dictionary are important, it is compulsory to normalized keyword weight. The maximum raw weight of every keyword among different owners are recorded as n-length list denoted as W_{max} where $W_{max} = (w_{i,1}^{raw}, w_{i,2}^{raw} \ldots\ldots)$ based on the vector W_{max} the normalized weight can be calculated as Wi,t = $\frac{\mathrm{W}_{i.t}^{raw}}{\mathrm{Wmax[j]}}$ **(4)** *Weight index generation. DO* obtain a secure weight index vector as $\widetilde{I}i.j = Ii.j \otimes Wi$ with high privacy protection strength. Where $\widetilde{I}i.j$ denoted as weight index vector of the document $Fi.j = (j \in 1, 2 \ldots n)$.

Balanced index tree (BIT) construction. BIT is a binary index tree structure used to construct index for efficient search in a system. We design a BIT-tree based on a secure weighted index $\widetilde{I}i.j$ and generate query vectors Q randomly. We used the "Greedy" method and bottom up strategy to pair similar nodes together. Based on probabilistic learning DO performs random searches to get the sum of matching scores for index and query vectors and sort the index according to descending order. The data structure of our index tree node i has 5 attributes: $\{ID, FID, Dv, Lch, Rch\}$ where ID stores the unique identifier for the node i. FID is the identifier of node i. If i is non-leaf node $FID = None$. D_v is the n-length vector of node i. Lch and Rch store the reference of left and right child node of i. We invoke the traditional algorithm to build BIT- tree on top of all document $di(i = 1, 2 \ldots m)$ and generate a unique identifiers $fFID$ of leaf node. If i is a leaf node it store document vector $\vec{D}di$ according to keyword dictionary and each dimension $i.\vec{D}$ is normalized as weighted index. If node i is the internal node and number of nodes is even,i. e., $2h$, assume that node i has t child nodes $(i1 \ldots it)$ then the vector is computed as $i.\vec{D}[v] = \max\{i1.\vec{D}[v], \ldots, it.\vec{D}[v]\}$ where $i = 1, \ldots, m$. if the number of input nodes is odd, i.e., $2h + 1$, create a parent node i_I for hth pair node, and then create a parent node i for i_I and the single node i_2. Finally, we obtain a binary index tree where the query complexity is close to *O(logN)* [18]

Build an Encrypted Index. After obtaining plaintext weighted indexes DO_i encrypt weight index tree τ with the secret key SK. where $SK = \{S, M_1, M_2\}$ to get an encrypted index tree $\widetilde{\tau}$. The index vector D_v of each node in a tree is split into two random vectors

$\{Dv'_1, Dv''_2\}$. Specifically if $S[j] = 0$, $Dv'_1[j]$ and $Dv''_2[j]$ will be equal to $Dv[j]$; else if $S[j] = 1$, $Dv'_1[j]$ is a random value $Dv''_2[j] = Dv[j] - Dv'_1[j]$ then each node encrypted index tree $\widetilde{\tau}$ contains two vector as follow $\widetilde{D_v} = \{M_1^T Dv'_1, M_2^T Dv''_2\}$. After encrypted, the vectors in all tree nodes DO_i send the encrypted index tree $\widetilde{\tau}$ to the CS. The construction of an encrypted index tree is completed. As the index tree is characterized by a set of nodes and set of pointers that specify all parent-child relationships, so the DO_i only encrypt the vector D_v carry in each node i, but it keeps all pointer constant. Therefore the encrypted and unencrypted index tree is isomorphic ($\tau \cong \widetilde{\tau}$).

Trapdoor Generation. To avoid the outflow of private information, the *DU* evaluated the trapdoors according to the search keyword set, which is also the encrypted form of a search request. When *DU* wanted to search the documents, they only needed to send the query request to the *CS*. DO_i generate query Q where $Q = (Q_1 \ldots\ldots Q_m)$ and build the *TD* using *SK* and send *TD* to *DU*. The same process is utilized to split Q into two random vectors $\{Q'_1, Q''_2\}$ the difference if $S[j] = 0$, $Q'_1[j]$ is a random value and $Q''_2[j] = Q[j] - Q'_1[j]$ else if $S[j] = 1$, $Q'_1[j] = Q''_2[j] = Q[j]$ where $j \in \{1, 2 \ldots .m\}$ Finally DO_i return encrypted Q as $TD = \left\{M_1^{-1}Q'_1, M_2^{-1}Q''_2\right\}$ and send *TD* to *CS*.

Search process of MERS-ML. (1) *Query preparation. DU* sends the query request to the *CS*. DO_i check whether the query is valid if yes then DO_i generate the *TD* and initiate search queries to the *CS*. If access control passes *CS* use the encrypted index tree $\widetilde{\tau}$ to search for index vectors that match the query vectors, and calculate the relevance score of an encrypted index vector in each tree node and trapdoor *TD.CS* return encrypted *top-k* documents to *DU* based on Rscore**(2)** *Calculate the relevance score.*

$$\begin{aligned}
&= Score : (D_v.TD) \\
&= \{M_1^T D'_{v_1} M_2^T D''_{v_2}\}.\left\{M_1^{-1}Q'_1 M_2^{-1}Q''_2\right\} \\
&= \left(M_1^T D'_{v_1}\right)^T.\left(M_1^{-1}Q'_1\right) + \left(M_2^T D''_{v_2}\right)^T.\left(M_2^{-1}Q''_2\right) \\
&= D'^T_{v1} M_1 M_1^{-1} Q'_1 + D''^T_{v_2} M_2 M_2^{-1} Q''_2 \\
&= D'_{v1}.Q'_1 + D''_{v2}.Q''_2 \\
&= RScore(D_v.Q)
\end{aligned} \tag{5}$$

The relevance score calculates from D_v and *TD* is exactly equal to that from D_v and this causes privacy leakage under known background model. To protect the trapdoors and keyword search under the known background model we should prevent the server from calculating the exact value by padding random noise into D_v and *TD*. To disturb the relevance score calculation during their generation. We generate random invertible matrix $(n + \ell) \otimes (n + \ell)$ the document vector will be extended to $(n + \ell)$ dimensions where ℓ is the random noise. We generated the index vector D_v of BIT-tree D_v is extended to $(n + \ell)$ dimension and the extended term $D_v[n + i]$, $i = 1 \ldots .\ell$ is set to a random number εi. Similarly, the query is also extended to $(n+\ell)$ dimensions and the elements are randomly set to 1 or 0. Thus the relevance score between query trapdoor and document vector is equal to $D_v \cdot Q + \sum \varepsilon j$ where $j \in \{i|Q[n + i] = 1\}$. The randomness of $\Sigma \varepsilon j$ ensures security under known background model. **(3)** *Search process of BIT-Tr*ee. BIT-tree used (GDFS) algorithm that can be executed to perform search on the index with high efficiency. The search algorithm is give below,

Algorithm 1 Search process of BIT-tree

```
Input:Root node of index tree $r$ and query $Q$
Output:The $Result-list$ which contain $top-k$ documents with corresponding $RScore$
1: if the node $i$ is not a leaf node then
2:     if Rscore $(D_v . Q) > kth$ score then
3:         GDFS $(i . high-relevance-child)$
4:         GDFS $(i . low-relevance-child)$
5:     else
6:         return
7:     end if
8: else
9:     if Rscore $(D_v . Q) > kth$ score then
10:        Update $Result-list$ and $kth$ score
11:    end if
12:    return
13: end if
```

4 Security Analysis

Data security. We use the symmetric encryption technique such as advanced encryption standard (AES) to encrypt the outsourced data. As long as the encryption key is not exposed the privacy of outsourced data is also guaranteed.

Index and query confidentiality. In the MERS-ML scheme the weight index vector and binary index tree is generated, So that it cannot leak any private information. In our tree index query vectors are generated randomly and search queries only return the secure inner products [9]. In our scheme, every data owner has their own encrypted keys and the ciphertexts is completely different for the same keyword in different data files while keeping searchable ability. More precisely the security of other data owners will not be compromised if the adversary cooperates with any DO_i and leak his important data content. Moreover, the security is further enhanced as the padding random noise [16] into data is difficult to figure the transformed matrices.

Trapdoor security. Introducing random noise in the query will generate different query vectors and receive different relevance score distribution with the same search request. [4, 20] The existence of random noise in query and data vector making it impossible for *CS* to distinguish two *TD* generated by any one query. The scheme ensured the unlinkability of *TD* in known background model.

Keyword search. In (MERS-ML) the document key is implemented with Attribute-based encryption (ABE) so that the adversary cannot gather any statistical information of keyword and documents. *DO* use the access control information to encrypt the documents and then store the encrypted documents in the *CS*, and *CS* not decrypt the encrypted ciphertext to obtain the document key. Moreover the weight of one keyword for different owners not the same, the adversary not determined that the two documents contain same keyword according to relevance score. In this manner, the security and privacy of key management is ensured.

5 Experimental Analysis

We implement the proposed scheme using Python in Window 10 operating system with Intel Core i5-7200U processor 2.50 GHz, 8 GB RAM and evaluated its efficiency on real-world dataset and compare it with MKRS-MO [10] EDMRS [4] we used the academic papers provided by Elsevier, http://elsevier.com/, including 30,000 papers and 90,000 different keywords, 600 academic conferences selected as data owners involving multiple domains All results represent the average of 1000 trails.

Index Tree Construction. After receiving binary index vectors *DO* construct a searchable binary index tree encrypt it and send it to the cloud server. The encryption process mainly depends on two multiplication of matrix and *n* dimension vector. For the construction of the tree, we used random searches in probabilistic and statistical sense. Since the BIT-tree is a balanced binary tree so the height of the tree is proportional to *log N*, and the search complexity is *O(logN)* (Where n is the number of nodes in the tree) which is used to retrieve top-k documents. Figure 2(a) shows that the time cost of generating an index tree is almost linear to the size of documents *D* when the size of keyword set is fixed. Figure 2(b) demonstrates that the size of keyword dictionary has great impact on the time cost of building index tree. Our scheme consumes less time than other existing scheme and is even more lightweight due to dimension reduction. Also, the time cost of construction is not ignorable overhead for the data owners it is one-time operation before data outsourcing.

Trapdoor Generation. Our scheme utilize the encryption process for the generation of trapdoors which follow vector splitting operation and two multiplication of $n \otimes n$ matrix. The time of generating trapdoors is extremely affected by the dimension of vector, and the time of generation of trapdoors (MERS-ML) is less than other schemes due to the dimensions of the different vector. Figure 3(a) shows that the time of generating trapdoor is highly affected by the size of the dictionary. Figure 3(b) shows the number of keywords in the query request hardly influences the overhead of trapdoor generation, because the dimension of Matrices and size of dictionary is always fixed.

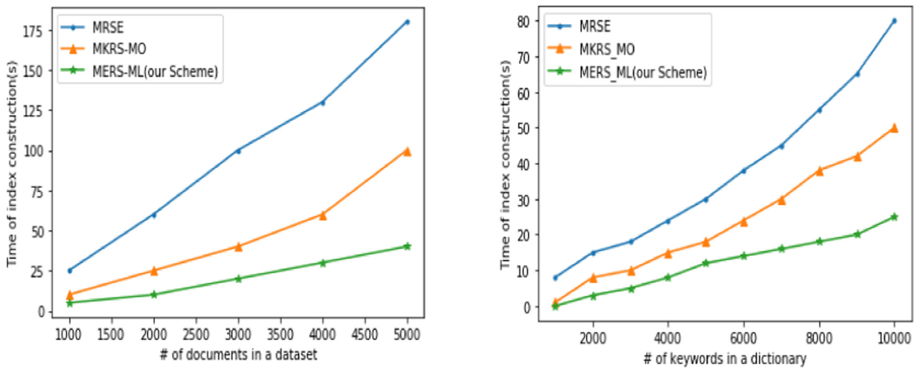

Fig. 2. Time cost of building index tree (a) For different size of dataset with the fixed dictionary u = 5000 (b) the fixed document dataset with different size of keyword dictionary n = 1000.

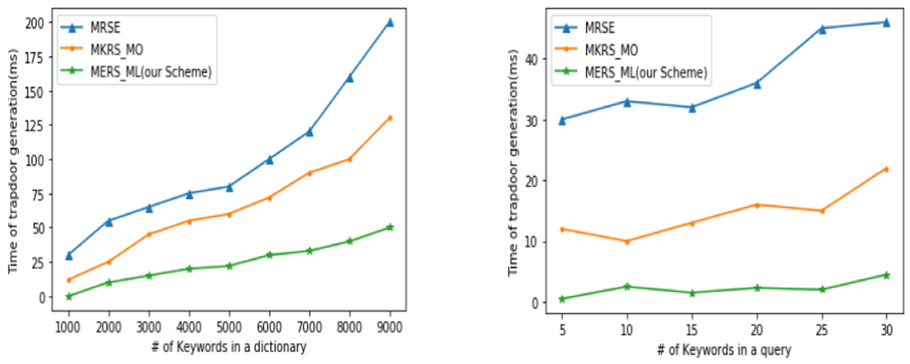

Fig. 3. Time cost of generating trapdoor (a) The same 5 query keywords within the different size of keyword dictionary (b) The Different number of query keywords within the same keywords dictionary, u = 5000.

Search Efficiency. This part of the experiment reveal the search efficiency of our scheme. The experiments on real-world dataset show that our results achieve near binary search and is superior to other comparison schemes. *DO* perform 10,000 random searches and get the sum of the matching scores of each index and all random query vectors. From Fig 4(a) we can see that the search efficiency of all the scheme increased with number of documents increases but our scheme achieve lower search time. Fig 4(b) shows the cost and search efficiency is improved by 5 times than the other comparison schemes in a different number of keyword search. By comparing the results we presume that when the size of the dataset increases the data features become sparser.

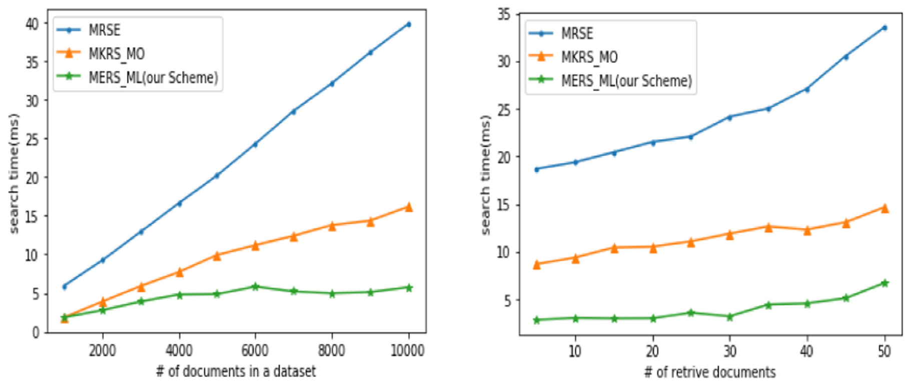

Fig. 4. Comparison of search operation (a) For different size of data set with the same size of query keywords q = 5 and keyword dictionary u = 5000 (b) For different number of retrieve documents with the same document and keyword dictionary n = 1000, u = 5000.

Moreover, the similarity between index and query vectors is mostly close to or equal to zero due to the sparseness of data features, which bring plenty of complications, and the construction of index tree is not a global order, so it is compulsory to traverse many nodes in the search, which show the limitation of the grouped balanced binary tree (MERS-ML).also the closer the number of random searches into infinity the higher the search efficiency of the index tree. Moreover, the maintenance cost of scheme based on BIT-tree is much lower than the other schemes. When data owner wants to add a new document into the *CS* we need to update the index tree by adding a new index leaf node in the index tree accordingly, and the search complexity of index tree is at least *O(logN)* times and for data updates it is at least *O(logN)* times, so that the total cost is *2O(logN)* [21] (where *N* is the number of node that index tree contained). Also, the update on an index is based on documents identifies, and no access to the content of documents required.

6 Conclusion

In this paper, we introduced secure and efficient (MERS-ML) scheme and conduct deep security and experimental analysis by combining Machine learning techniques. To solve the problem of the quality of deferent documents of multiple owner model we cluster index vectors into multiple indexes and divide the keyword dictionary into multiple sub-dictionaries by using k-means clustering approach. Besides, our Scheme proposes principle component analysis (PCA) to avoid the curse of dimensionality, caused by big data sparsity and reduce the dimensions of index vector which improve the efficiency of secure (KNN) algorithm. Last but not least, we constructed a balanced index tree (BIT-tree) generated by sufficient amount of random searches and follow Greedy depth first algorithm to obtain better computational complexity close to *O(logN)*.The experiments on real world dataset show that our scheme is secure against threat models and prove the flexibility and efficiency of the system.

Acknowledgment. This work is partially supported by the Fundamental research Funds for the Central Universities (NO. 30918012204).

References

1. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE Symposium on Security and Privacy. S & P 2000, pp. 44–55 (2000)
2. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. J. Comput. Secur. **19**, 895–934 (2011)
3. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
4. Xia, Z., Wang, X., Sun, X., Wang, Q.: A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **27**, 340–352 (2015)

5. Sun, W., et al.: 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, pp. 71–82 (2013)
6. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: 2010 IEEE 30th International Conference on Distributed Computing Systems, pp. 253–262 (2010)
7. Fu, Z., Wu, X., Guan, C., Sun, X., Ren, K.: Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Trans. Inf. Forensics Secur. **11**, 2706–2716 (2016)
8. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: 2010 Proceedings IEEE INFOCOM, pp. 1–5 (2010)
9. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **25**, 222–233 (2013)
10. Guo, Z., Zhang, H., Sun, C., Wen, Q., Li, W.: Secure multi-keyword ranked search over encrypted cloud data for multiple data owners. J. Syst. Softw. **137**, 380–395 (2018)
11. Zhang, W., Xiao, S., Lin, Y., Zhou, T., Zhou, S.: Secure ranked multi-keyword search for multiple data owners in cloud computing. In: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 276–286 (2014)
12. Miao, Y., et al.: VCKSM: Verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings. Pervasive Mob. Comput. **40**, 205–219 (2017)
13. Li, J., Lin, Y., Wen, M., Gu, C., Yin, B.: Secure and verifiable multi-owner ranked-keyword search in cloud computing. In: Xu, K., Zhu, H. (eds.) WASA 2015. LNCS, vol. 9204, pp. 325–334. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21837-3_32
14. Smith, L.I.: A tutorial on principal components analysis (2002)
15. Peng, C.: Distributed K-Means clustering algorithm based on fisher discriminant ratio. j. Jiangsu Univ. Natl. Sci. Ed. **35**, 422–427 (2014)
16. Wong, W.K., Cheung, D. W.-l., Kao, B., Mamoulis, N.: Secure kNN computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pp. 139–152 (2009)
17. Delf, H., Knebl, H.: Introduction to Cryptography: Principles and Applications. Springer, Berlin (2007)
18. Knuth, D.: The Art of Computer programming, Vol 3: sorting and searching, 2nd edn. Addison-Wesley Publ. Co, Boston (1998)
19. Salton, G., Wong, A., Yang, C.-S.: A vector space model for automatic indexing. Commun. ACM **18**, 613–620 (1975)
20. Ballard, L., Kamara, S., Monrose, F.: Achieving efficient conjunctive keyword searches over encrypted data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005). https://doi.org/10.1007/11602897_35
21. Chen, K., Lin, Z., Wan, J., Xu, L., Xu, C.: Multi-owner secure encrypted search using searching adversarial networks. In: Mu, Y., Deng, R.H., Huang, X. (eds.) CANS 2019. LNCS, vol. 11829, pp. 184–195. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31578-8_10