Multi-keyword Ranked Search with Fine-grained Access Control over Encrypted Cloud Data

Jingyu Lei[1,2, a], Jiao Mo[1, b]

[1]School of Science, Beijing University of Posts and Telecommunications, Beijing 100876, China;

[2]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China.

[a]leijingyu2015@163.com, [b]pridenope@gmail.com

Keywords: multi-keyword search, access control, privacy-preserving, cloud computing

Abstract. With the development of cloud computing, people have become accustomed to using these cheap and convenient services provided by the cloud server for data storage and processing. For protecting data privacy from leaking information to unauthorized users, probably including the cloud service provider (CSP), the data must be stored in an encrypted form before outsourcing, which leads to some traditional data services being unavailable, for example, data search. Therefore, to design a privacy-preserving data search scheme is an extremely challenging problem. Meanwhile in order to keep data from using (and even searching) by unauthorized users, it also needs to make sure that the search scheme could achieve fine-grained access control on data users, according to the access policies made by the date owner (DO).In this paper, considering both the control of user's access authority and the privacy protection in data search, we have designed a multi-keyword searchable scheme with access control over encrypted cloud data. By using the technology of inverted indexes, we can rapidly filter out data files that users own authority to access, while methods of coordinate matching and secure inner product computation have also been introduced to realize the privacy-preserving multi-keyword search. Through analysis of privacy and efficiency, we prove that our scheme achieves the requirements of privacy protection under two different adversary models, and that it has relatively high efficiency from the perspective of time complexity.

1. Introduction

With the advent of cloud computing, more and more enterprises and individuals outsource data to the cloud, and use services provided by CSP, such as data storage, information search, or outsourced calculation [1]. However, cloud computing has also brought new security risks with data far from DO's physical control after stored in the cloud server, so it’s difficult to protect their confidentiality and privacy. For some sensitive datas, the leak will cause huge losses. Therefore, many data owners choose to encrypt datas before outsourcing, rather than in the form of plaintext. It is a challenge to design a privacy-preserving data searchable scheme.

In general, users always would like to select out the most relevant documents through a simple query in massive files. Under the commercial cloud computing model of "pay as you use" [2], it becomes more important to achieve accurate and efficient data search. In addition, even if the dataset is outsourced by same DO, the attributes of the user community of DO target are likely to be different (for example, for the outsourced dataset of an enterprise, the settings of access permission for different departments, or different levels of the same department, are not the same). This requires the search service only to return the most relevant documents that user own authority to access responding to his query, according to the access authorities set by DO for different files. On the one hand, this can realize a more fine-grained access control of DO, on the other hand, this also could save the computational cost of CSP to some extent (obviously, it is meaningless for files that users don’t have authority to access to carry out the search operation).

In 2010, Wang [3] proposed a highly efficient ranked searchable scheme with single keyword, which can get discriminating search results when users are searching on the encrypted dataset. However, this scheme couldn’t be implemented if user’s query contains multi-keyword, and it is

clear that the latter can improve the accuracy of the query. Later on the basis of Wang, Ning Cao [4] proposed a multi-keyword searchable scheme over encrypted data through using “coordinate matching” to calculate scores of the query to obtain ranked search results, which has been the most mature multi-keyword searchable scheme over encrypted data by far. Then on the basis of Ning Cao [4], through improving search algorithm, a lot of work has been done to achieve a more accurate and efficient multi-keyword search, which are also the foundation of our scheme in this paper. However, these models all assume that the user and DO are in completely trustful relationship, in which user could search and access all outsourced files. Once the user generates the trapdoor for the search operation under the authorization of DO, then user is able to search all the files he is interested in and download them. That is, these schemes couldn’t achieve fine-grained access control.

Some schemes [5], [6] of attribute based encryption (ABE) designed with access control policy are based on the user’s attributes, in which DO can set the corresponding access control structures related to attributes for each file. Only when the user’s attributes match the access structure of one document, then he could successfully decrypt it. This method helps user use access control to manage remote encrypted data to a great degree. Based on the similar ideas, Shen et al. [7] designed a scheme of keyword search with access control, however the efficiency of HPE they used was general, and meanwhile the scheme was not considered in ranking results of search. In addition, the general ABE schemes only provided download service, so it was not required for CSP to filter the files set in advance. But in our multi-keyword searchable scheme with access control, the efficiency of search will be directly related to user’s experience. Therefore, we just learn how to design the access structures from these ABE schemes.

In this paper, we have considered both the control of user's access authority and the privacy protection in data search. The basic structure of our scheme is composed of two parts of algorithm: filter operation and search operation. Firstly, through the user's attributes authenticated by the certification center, CSP filters out the files that user has authority to access, and calculates the similarity scores of file indexes and query in the collection of selected documents in accordance with the query, then returns the ranked results to user. In order to update attributes and reduce computation load of user's decryption in a more convenient way, in system we have selected a trusted authentication center (CA), which is responsible for the authentication of attributes. This also simplifies our protocol to a certain extent. For different threat models, we have designed two schemes. In scheme 1, we firstly develop the respective access policy for each file, and use the inverted search to efficiently filter out files that users could access. Further, in order to reduce the cost of communication, CSP returns *top-k* files of the outsourced dataset to user, according to the parameter *k* set by user. In scheme 2, in order to protect the privacy of attributes, the blind of the attributes’ values (including access policies of files and user's attributes) is carried out by using hash functions and pseudo random functions based on scheme 1, which can achieve the filter operation without leaking attributes. Above all, our contributions are as follows,

1) We propose one kind of efficient and concise scheme about multi-keyword ranked search with fine-grained access control, which could save the computational cost of CSP through two layer algorithm-- firstly filter operation, then rank operation.

2) With the blind of access policies of files set by DO and attributes of users, our scheme protects the privacy of the users' attributes.

3) Through analysis of privacy and efficiency, we prove that our scheme meets the requirements of privacy protection defined under two different adversary models, and that it has indeed introduced low overhead on computation and communication with a relatively higher efficiency from the perspective of time complexity.

2. PROBLEM FORMULATION

2.1 System Model

We have designed two different system models based on whether the user is required to protect his attributes which are very sensitive to user, such as job, age, health condition, etc. The scheme 1

doesn't consider protecting the privacy of user's attributes, while the scheme 2 built on the basis of Scheme 1 considers it. Simply, the description of the system model selects the scheme 1 as an example. A cloud data hosting service involves four different entities, as illustrated in Fig. 1: the data owner, the data user, and the cloud server and the certificate authority. The data owner has a collection of data documents *F* to be outsourced to the cloud server in the encrypted form *C*. To enable the searching capability and set the documents access authority over *C* for effective data utilization and access control, before outsourcing, the data owner will first build a searchable index *I* and access policy *P* from *F*, then encrypt *I*, and finally outsource all the encrypted indexes, policies and the encrypted document collection to the cloud server. To search the document collection for *t* given keywords, a user needs to send a search request to CSP, which should contain two aspects of information: a corresponding trapdoor *TR* through search control mechanisms and an attribute collection which has been certificated by certificate authority (CA). Upon receiving search request from a data user, CSP should firstly filter out the files that user has authority to access according to the his attributes, then perform the search operation in the collection of these files, and finally return the most relevant *top-k* ranked results to user.

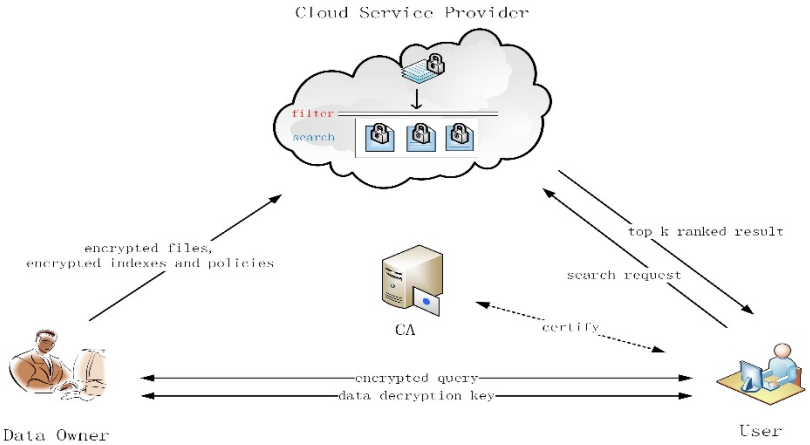

Fig. 1Architecture of the search over encrypted cloud data in scheme 1

2.2 Threat Model

The cloud server is considered as "honest-but-curious" in our model, which is consistent with related works on cloud security [8], [9]. Specifically, the cloud server acts in an "honest" fashion and correctly follows the designated protocol specification. However, it is "curious" to infer and analyze data (including index) in its storage and message flows received during the protocol so as to learn additional information. Based on the different degree of privacy protection of data users, we consider two kinds of threat models, as follows:

1) Data users don't need to protect the privacy of their attributes: data users just don't want the CSP to know their detailed search content, and even the subject of its search.

2) Data users want to protect their identity attributes: not only protect the privacy of the search content, data owners are also not willing to let the cloud service providers know their identity attributes (i.e. sex, age, job, etc.), because these informations are likely to be users' sensitive privacy.

2.3 Design Goals

· *Multi-keyword Ranked Search*: To design schemes which allow multi-keyword query and provide result similarity ranking for effective data retrieval, instead of returning undifferentiated results.

· *Access Control Search*: In order to let the data users can only carry on the multi-keyword search in the documents, when the users' identity attributes accord with the access policy of the relevant documents, we need to establish a reasonable access control mechanism.

· *Privacy-Preserving*: To prevent the cloud server from learning additional information from the dataset, and to meet privacy requirements specified in section 2.6.

· *Efficiency*: Above goals on functionality and privacy should be achieved with low communication and computation overhead.

2.4 Notations

· *F*–the plaintext document collection, denoted as a set of *N* data documents *F*= $(F_1, F_2 \dots \dots F_N)$.

· *C*–the encrypted document collection stored in the cloud server, denoted as *C*= $(C_1, C_2 \dots \dots C_N)$.

· *P*–the policy collection associated with document collection*F*, denoted as *P* =$(P_1, P_2 \dots \dots P_N)$.

· *A*–a collection of attributes(i.e. sex, age, job, etc.), denoted as$A = (A_1, A_2 \dots \dots A_m)$.

· *D*–the dictionary, i.e., the keyword set consisting of p keywords denoted as$D = (d_1, d_2 \dots \dots d_p)$.

· *I*–the searchable index, denoted as $I = (I_1, I_2 \dots \dots I_N)$ where each subindex I_i is built forF_i.

· $\widetilde{D}$–the subset of *D*, representing the keywords in a search request.

· *TR*–the trapdoor for the search request$\widetilde{D}$.

2.5 Preliminary

· *Coordinate Matching*

As a hybrid of conjunctive search and disjunctive search, "coordinate matching" [10] is an intermediate similarity measure which uses the number of query keywords appearing inthe document to quantify the relevance of that document tothe query. When users know the exact subset of the datasetto be retrieved, Boolean queries perform well with the precisesearch requirement specified by the user. In cloud computing,however, this is not the practical case, given the huge amountof outsourced data. Therefore, it is more flexible for users tospecify a list of keywords indicating their interest and retrievethe most relevant documents with a rank order.

· *Secure Inner Production*

This algorithm was proposed initially in the secure calculation of *kNN* [11]. The model extracted from the specific environment can be described as follows: we assume that there are two participants, player 1 and player 2, while player 1 has a vector $\vec{I} = (I_1, \dots, I_p)$ and player 2 has a vector $\vec{Q} = (q_1, \dots, q_p)$. Now it is required to calculate the inner production of $\vec{I}$ and $\vec{Q}$ by a third-party server C, but the two players are not willing to reveal their privacy to C, so they firstly share a key $sk = \{s, M_1, M_2\}$, where *s* is a randomly generated *p*-bit vector donated as $s = \{0,1\}^p$, and $M_1$and M_2 are two $P \times P$ randomly generated invertible matrices.

Firstly, player 1 uses *sk* to encrypt vector $\vec{I}$, and the method is as follows:

1) According to *s*, $\vec{I}$ is split into two random vectors as $\{\vec{I}^{(1)}, \vec{I}^{(2)}\}$. Note here that *s* functions as a splitting indicator. Namely, if the q-th bit of *s* is 0,

$$\vec{I}^{(1)}[q] = \vec{I}^{(2)}[q] = \vec{I}[q],\ q \in [1,\ p]$$

If the q-th bit of *s* is 1, $\vec{I}^{(1)}[q] + \vec{I}^{(2)}[q] = \vec{I}[q]$

2) Using $\{M_1, M_2\}$, the split data vector pair $\{\vec{I}^{(1)}, \vec{I}^{(2)}\}$ is encrypted as $\{M_1^T \cdot \vec{I}^{(1)}, M_2^T \cdot \vec{I}^{(2)}\}$, which is marked as $E(\vec{I})$. Then player 1 sends $E(\vec{I})$ to C.

Secondly, player 2 similarly use *sk* to encrypt vector $\vec{Q}$, and the method is as follows:

1) According to *s*, $\vec{Q}$ is split into two random vectors as $\{\vec{Q}^{(1)}, \vec{Q}^{(2)}\}$. Note here that *s* functions as a splitting indicator. Namely, if the q-th bit of *s* is 0,

$$\vec{Q}^{(1)}[q] + \vec{Q}^{(2)}[q] = \vec{Q}[q],\ q \in [1,\ p]$$

If the q-th bit of *s* is 1, $\vec{Q}^{(1)}[q] = \vec{Q}^{(2)}[q] = \vec{Q}[q]$

2) Using $\{M_1^{-1}, M_2^{-1}\}$, the split data vector pair $\{\vec{Q}^{(1)}, \vec{Q}^{(2)}\}$ is encrypted as $\{M_1^{-1} \cdot \vec{Q}^{(1)}, M_2^{-1} \cdot \vec{Q}^{(2)}\}$, which is marked as $E(\vec{Q})$. Then player 2 sends $E(\vec{Q})$ to C.

Thirdly, C calculates

$$E(\vec{I}) \cdot E(\vec{Q}) = \{M_1^T \cdot \vec{I}^{(1)}, M_2^T \cdot \vec{I}^{(2)}\} \cdot \{M_1^{-1} \cdot \vec{Q}^{(1)}, M_2^{-1} \cdot \vec{Q}^{(2)}\} = \vec{I} \cdot \vec{Q}$$

3. Scheme Description

Scheme 1 *:·*

· *Setup* (1^λ): The date owner inputs a secure parameterλ , outputs a key *sk*=$\{s, M_1, M_2\}$, in which *s* is a *p*-bit binary vector denoted as$\{0,1\}^P$, andM_1, M_2 are two $p \times p$ random reversible matrixes.

· *Upload (F, D, sk)*:Do uploads the date files being outsourced to the CSP, which can provide appropriate search service for the legitimate users. We divide the upload process into three steps:

1) Indexes Generation: DO extracts keywords for each file in the files set, and generates an index set$I = \{I_1, I_2, \ldots, I_N\}$ for the files set F according to the dictionary D shared within system, in which $I_i$is a p-bit binary vector $\{0,1\}^P$, while I_i[q]=1 indicates that the keyword $d_q$appears in the file F_i, andI_i[q]=0 indicates that $d_q$doesn't appear in the file F_i.
2) Index encryption: DO uses the *sk* to encrypt the index set *I*, as shown as $I \xrightarrow{sk} \mathrm{E}(I)$. In which the encryption algorithm is consistent with that encryption algorithm of security inner product calculation about vector$(I_1, \ldots I_p)$, and we denote the encrypted indexes set as E(*I*).
3) Policy formulation: DO sets access policies for each file being outsourced, which can be constructed with dendrogram as shown in *Fig. 2*.

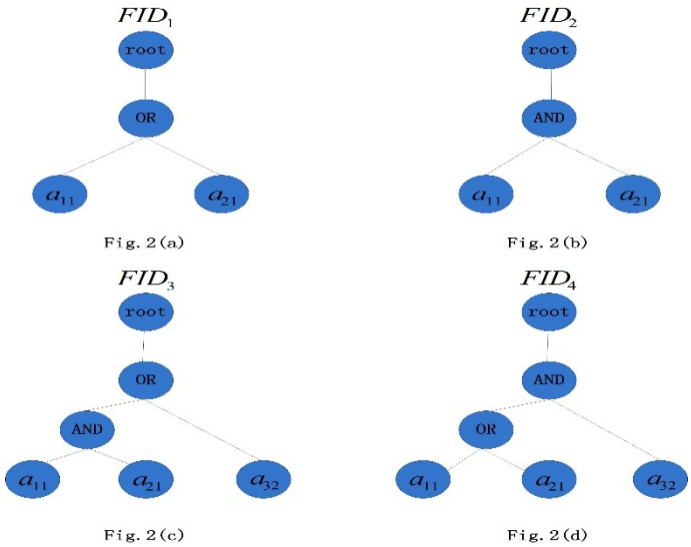

Fig. 2: The four basic dendrograms

In which a_{ij}denotes one attribute value, and FID_idenotes the file's ID. After generating the policy treeP_i for each fileF_i, DO uses symmetric cryptography to encrypt the files set *F*as *C*. Then DO uploads the 3-tuple in the form as $\{C, \mathrm{E}(I), P\}$ to the CSP.

· *Pretreatment (P)*: Based on these access policy trees of each file, CSP generates a Attribute-File-List denoted as AFL. AFL is made up of two sets of data, as shown in Table 1.

Table 1 Attribute-File-List

Access Policy	File's ID
a_{11}	FID_1
a_{21}	FID_1
$a_{11} \wedge a_{21}$	FID_2
$(a_{11} \wedge a_{21}) \vee a_{32}$	FID_3
$(a_{11} \vee a_{21}) \wedge a_{32}$	FID_4
……	……

· *TRGen($\widetilde{D}$)*: The DO inputs the words set $\widetilde{D}$ consisted of *t* keywords to generate the query *Q* which is a p-bit binary vector $\{0,1\}^P$, and $Q[q] = 1$ $(1 \le q \le p)$ indicates that the keyword $d_q$appears in the query, while$Q[q] = 0$ indicates that $d_q$doesn't appear in the query. We assume that the data owner has the ability to authenticate user's identity, by which he could share the secret key *sk* with users through the search control mechanism (i.e. broadcast encryption) . In this way, the user can use *sk* to encrypt *Q* to generate trapdoor, in accordance with method of the secure inner product computing about the vector $(q_1, \ldots q_p)$. As is shown as $Q \xrightarrow{sk} TR$.

· *Search (U_A,T_R)*: The search process can be composed of the following three steps:

1) Before sending a search request, the user sends his attributes denoted as U_A to the *CA*(Of courseU_A can also be expressed as the interaction or union of a few attributes, as shown in the left column in Table. 1). After identifying the authenticity ofU_A, the *CA* will sign on it. We denote the signature as$Sign$（U_A）,and return it to user.
2) The user sends the search request denoted as $\{U_A, Sign（U_A）, TR\}$ to the cloud server.
3) Having received the search request sent from the user, the cloud service provider first verifies the authenticity of U_A according to the $Sign$（U_A）, as shown in the following equation.

$$Ver(Sign(U_A), PK_{CA}) \rightarrow \begin{cases} true \\ false \end{cases},$$

If the authenticity of U_A is validated, according to the prior construction of the *AFL*, CSP could lookup files that the user owns authority to access, and extract the corresponding encrypted indexes set E($\tilde{I}$) representing a subset of E(I).Then, CSP uses the trapdoor sent by the user and E($\widetilde{I_l}$)（E($\widetilde{I_l}$)∈ E($\tilde{I}$)）to do inner production, then gets the similarity scores, as shown as $E(\widetilde{I_l}) \cdot TR=\{M_1^T E(\widetilde{I_l})^{(1)}, M_2^T E(\widetilde{I_l})^{(2)}\} \cdot \{M_1^{-1} TR^{(1)}, M_2^{-1} TR^{(2)}\} = \widetilde{I_l} \cdot Q$

Through these three steps, we can get the ranked results, according to the correlation degree of the search request. Later, the cloud service provider can return the *top-k* files that are the most consistent with the search request to the user based on the parameters *k* provided in advance by the user.

Scheme 2:

Through the scheme 1, we find that the whole process not only requires the data owner to outsource these access policies, but also requires the user to provide his corresponding identity attributes with authentication information. While in real life, users don't want to expose their own identity attributes in many situations. To solve this problem, we use the property of hash function and pseudo random function to modify the corresponding protocols in scheme 1, as shown in Fig. 3.

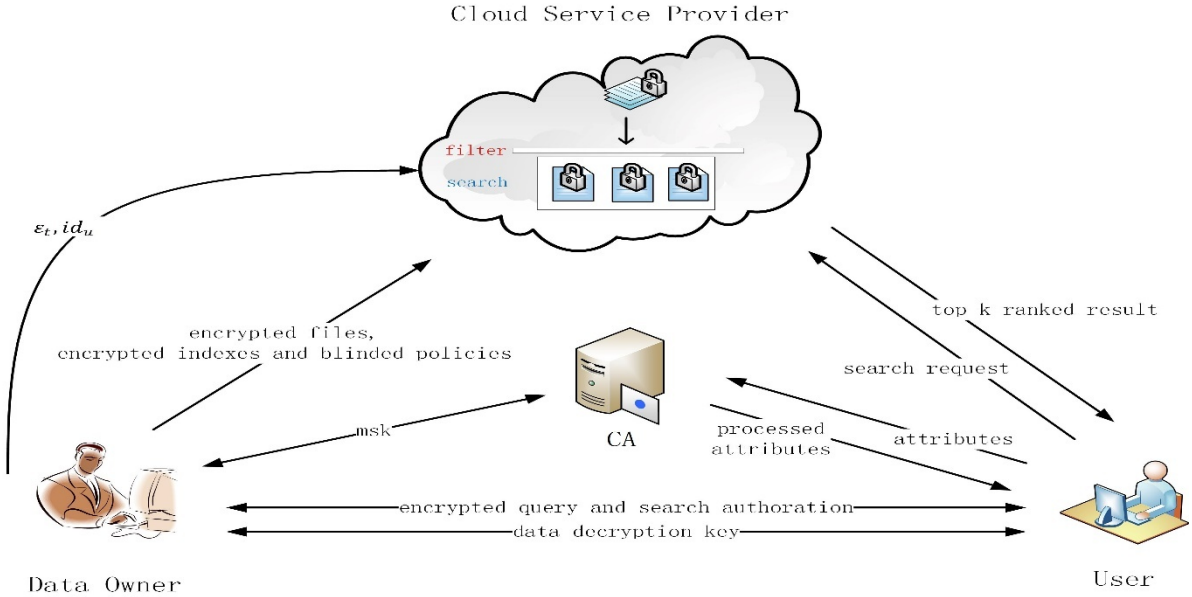

Fig. 3Architecture of the search over encrypted cloud data in scheme 2

In the Fig. 3, we can see that the function of each entity is similar to that one of the scheme 1, while the difference is that the process of protocols is more complex to achieve to protect the privacy of user's identity attributes. Specifically, as is shown with the following five steps:

· *Setup* (1^λ):Similar to the scheme 1, the data owner generates a key *sk* for encrypting indexes according to security parameterλ. Then data owner selects two secure pseudorandom functions φ andϕ, a random variable ε_t ($\varepsilon_t$is randomly generated each time for different users or the same user about different queries), as well as the system master key *msk* shared with CA (we assume that the CA is completely credible). Meanwhile, data owner sends the tuple (ε_t, id_u) to the CSP.

· *Upload (F，D，sk，φ，msk)*: Same as the scheme 1, the data owner generates the index and formulates the access policy for each file, and encrypts files and indexes before outsourcing the data

files, in which the algorithm used is consistent with the scheme 1. The difference is that the data owner will no longer upload the access policies of files in plaintext, but use the pseudo random function φ to generate blinded attributes for all the attribute values that appear in the access policies.

$$\{r_x\}_j \leftarrow \varphi(k||j, \{a_x\}_j),$$

In which k is the master key *msk*; and j is the subscript of the file F_j, while $\{a_x\}_j$ denotes a collection of attribute values in the access policy P_j set for the file F_j (For example, attribute values a_{11},a_{21},a_{32} appear in the access policy $(a_{11} \wedge a_{21}) \vee a_{32}$, so $\{a_x\}_j = \{a_{11}, a_{21}, a_{32}\}$), and$\{r_x\}_j$denotes a collection of the each attribute value in $\{a_x\}_j$ after mapped by the pseudo random function φ. Then the data owner recombines $\{r_x\}_j$ with *AND*and*OR* in logical structures according to the access policies, which is denoted asr_j. Next, he uploads the encrypted files set, the encrypted indexes set and the collection of the access policies (denoted as R, $R = \{r_1, r_2, \ldots, r_N\}$) which have been processed with pseudo random function to CSP. Obviously, the adversary can't guess out the access policies in plaintext when he doesn't know the master key *msk*.

· *Pretreatment(p)*:The preprocessing stage is similar to the scheme 1. The difference is that the CSP no longer receives access policy trees in plaintext, but access structures formed by random values. We denote the collection of access structures corresponding to all the files as $R = \{r_1, r_2, \ldots, r_N\}$ when r_j corresponds to the file F_j, then we can construct a list in which policies correspond to files with the same method in accordance with the scheme 1. We denote this list as *RFL*.

· *TRGen* $(\widetilde{D})$: The generation of trapdoor is in agreement with the scheme 1.

· *Search* (U_A, T_R):The search process is different with scheme 1. The four steps are as follows:

1) First, the user sends the search request to DO. After giving the consent, DO sends $H(ID_i, \varepsilon_t)$ to user whose identity is ID_i as a search authorization, while $\varepsilon_t$denotes a random variable.
2) The user sends his attributes set $U_A$to CA. After the certification of CA,U_A is mapped to a set of random values by means of a pseudo random function by CA, as shown as

$$\{U_{A_1}, U_{A_2}, \ldots, U_{A_m}\} \xrightarrow{\varphi、msk} \{r_1', r_2', \ldots, r_m'\},$$

   In which $r_x' = \varphi(k||i, U_{A_x})$, $x \in [1, m]$, m denotes the total number of attribute values of one user, and i denotes the subscript of attribute domain A_i corresponding with the attribute value U_{A_x}. We remember $U_A' = \{r_1', r_2', \ldots, r_m'\}$, then CA sends U_A' to user.
3) After receivingU_A', through calculating$L = \phi(H(ID_i, \varepsilon_t), U_A')$, the user generates a new tuple that we remember as L.In which $L = \{L_1, L_2 \ldots, L_m\}$, $L_x = \phi(H(ID_i, \varepsilon_t), r_x')$,$x \in [1, m]$, finally, the user sends the trapdoor and L as the search request to CSP.
4) The L' is calculated as $L' = \phi(H(ID_i, \varepsilon_t), R)$by using the tuple （$\varepsilon_t$, id_u） received in the setup phase and the access control policies R processed by the data owner,

Then, CSP compares the elements in the same location of $L'$and L. if matching, it can explain the data owner meets the access policies in the corresponding *RFL*. Similarly, CSP first selects the files conforming to access policies, and then performs the search operation following the multi-keyword search model, finally returns the *top-k* files that are the most consistent with the search request to user.

4. Security Analysis and Performance Evaluation

4.1 Privacy Analysis

Data privacy: For the privacy protection of data files, we use the traditional symmetry secret-key encryption technology to encrypt data sets and then upload it to the CSP, so as long as the secret-key for encryption is secure, data privacy will not be leaked. We do not consider this in our scheme.

Index privacy: The indexes are encrypted by the key *sk*, and the security of this encryption algorithm has been proved to be safe in the known ciphertext model [11], so as long as the *sk* used to encrypt indexes is be leaked, the privacy of the indexes will be safe.

Trapdoor unlinkable: In the terms of trapdoor, because it is generated from the encrypted query made by the user and the encryption algorithm is similar with that one about indexes. In the same way we can know the privacy of trapdoor is also safe in the known ciphertext model. Meanwhile, because of the randomness of segmentation stage of the query, even for the same two queries generated by the same user, their corresponding trapdoors are different in form, however the attacker couldn't distinguish whether any two trapdoors correspond with one query.

Attribute privacy: In scheme2, the data owners will no longer upload the access policies to CSP in the plain text, but they use the pseudo random function to map attribute values of the access policies to the meaningless random values for the attackers, then through the "AND "and "OR " to generate new access policies to upload. Similarly, User could add attribute values handled by pseudo random function into the search request in the same way, thus CSP will be able to filter out files when access policies and user's attributes have been blinded. Here, the security of attribute values is equivalent to the safety of the pseudo random function, so we believe in a secure situation of pseudo-random function, the privacy of attribute values is also secure.

4.2 Efficiency

We focus on the analysis of the entire scheme from the perspective of time complexity.

Encrypted Index Construction: From the algorithm of the construction of encrypted index, we know that the index encryption of one file is composed of p-step random segmentation and $2 \times p^2$-step multiplication, so the time complexity of the algorithm to encrypt index is $O(p^2)$. We assume that DO has N files in total, so the time complexity of DO to encrypt the total indexes set is $O(N\times p^2)$. It is thus obvious that the time of the index encryption will increase with the increase of numbers of files and the length of the dictionary. In general, the length of the dictionary will be bigger than the numbers of files. Considering the influence of the length of dictionary to the time complexity is in quadratic level. Therefore, the impact of the increase in the length of the dictionary will be much greater than the increase in the number of files on the encryption time.

Trapdoor Generation: Similarly with the index encryption, because the user only generates one query at a time, the time complexity of the corresponding trapdoor based on the query is$O(p^2)$. The time is only related to the length of the dictionary. The longer the length of the dictionary is, the more time will be needed to generate the trapdoor..

File Filter: Without loss of generality, we take scheme 1 as an example, because of the difference between two schemes just is using the meaningless random values to replace those meaningful attribute value, while the computational efficiency of the comparison operation is consistent of the two schemes. Assuming that it takes t moments to compare one attribute of the user with one access structure in the left column of the AFL. Then it takes mt moments for us to get all of the list of selected files' ID after traversing the whole in the left column of the AFL once, in which m denotes the total number of different access policies of DO, so the time complexity of the filter operation is $O(m)$. However, if we don't take the method of inverted index to check access structures of files in order, it takes *Nt* moments, in which the N is the total numbers of uploaded files, so the time complexity of the filter operation is $O(N)$. Obviously $m<N$, that is to say, the application of the inverted index can improve the efficiency of the filter operation, and the time consumption of the filter operation is only related to the total numbers of different access policies formulated by DO.

Search: According to our scheme, the search operation is composed of two processes, similarity scores computation and ranking the scores. Among that the similarity scores is obtained by calculating inner production of trapdoor and index. We learn that the whole calculation process is composed of $2p$-step multiplication operations and $2(p-1)+1$-step addition operations. Next we assume that the number of selected files is $N'(N'<=N)$, so the time complexity of ranking operation is $O(N'\log N')$ according to the quick sorting algorithm. In summary, the total time complexity of search operation is $O(p+ N'\log N')$. We can find that the time consumption of

the search operation is both related to the numbers of selected files and the length of the dictionary. The more selected files and the bigger length of the dictionary, the more time it will take.

In summary, we can list the time complexity at each stage of our scheme, as shown in Table. 2:

Table 2 Time Complexity at Each Stage

Stage	Time Complexity
Encrypted Index Construction	$O(N\times p^2)$
Trapdoor Generation	$O(p^2)$
File Filter	$O(m)$
Search	$O(p+ N'\log N')$

It is found that the time complexity is relatively small for the user to compute in the phase of trapdoor generation, which also proves the feasibility of our scheme. And even if the user uses a lightweight device, our scheme also has a fairly good usability.

5. Conclusions

In this paper, considering both the control of the user's access rights and the privacy protection in data retrieval, we have designed two multi-keyword search schemes over the cloud encrypted data to achieve fine-grained access control. We develop the respective access policy for each data file in the project firstly, and by using the inverted search technology, we can quickly screen out the data files that uses own rights to access with the improvement in efficiency. Under two different threat models, the two schemes that we have proposed are also based on coordinate matching and secure inner product computation technology, which are introduced to realize the privacy preserving multi keyword retrieval. Meanwhile, through using hash functions and pseudo random functions to deal with the attributes of the plaintext, we achieve the purpose of the security of the policy. Through security analysis, privacy protection guarantees of the proposed schemes have been proved, and experiments on the real-world dataset further show the proposed schemes indeed introduce low overhead on computation and communication, with a relatively higher efficiency.

Acknowledgments

This work is supported by NSFC (Grant Nos. 61300181, 61502044), the Fundamental Research Funds for the Central Universities (Grant No. 2015RC23).

References

[1]. Armbrust M, Fox A, Griffith R, et al. A view of cloud computing [J]. Communications of the ACM, 2010, 53(4): 50-58.

[2]. Cao N, Yu S, Yang Z, et al. LT codes-based secure and reliable cloudstorage service[C]//INFOCOM, 2012 Proceedings IEEE. IEEE, 2012:693-701.

[3]. Wang C, Cao N, Li J, et al. Secure ranked keyword search over encrypted cloud data[C]//Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on. IEEE, 2010: 253-262.

[4]. Cao N, Wang C, Li M, et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data [J]. Parallel and Distributed Systems, IEEE Transactions on, 2014, 25(1): 222-233.

[5]. Goyal V, Pandey O, Sahai A, et al. Attribute-based encryption for fine-grained access control of encrypted data[C]//Proceedings of the 13th ACM conference on Computer and communications security. Acm, 2006: 89-98.

[6]. Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption[C]//Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007: 321-334.

[7]. Shen Z, Shu J, Xue W. Keyword search with access control over encrypted data in cloud computing[C]//Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of. IEEE, 2014: 87-92.

[8]. Yu S, Wang C, Ren K, et al. Achieving secure, scalable, and fine-grained data access control in cloud computing[C]//INFOCOM, 2010 Proceedings IEEE. Ieee, 2010: 1-9.

[9]. Wang C, Wang Q, Ren K, et al. Privacy-preserving public auditing for data storage security in cloud computing[C]//INFOCOM, 2010 Proceedings IEEE. Ieee, 2010: 1-9.

[10]. Witten I H, Moffat A, Bell T C. Managing gigabytes: compressing and indexing documents and images [M]. Morgan Kaufmann, 1999.

[11]. Wong W K, Cheung D W, Kao B, et al. Secure kNN computation on encrypted databases[C]//Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. ACM, 2009: 139-152.