This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

Semantic-aware Searching over Encrypted Data for Cloud Computing

Zhangjie Fu Lili Xia Xingming Sun Alex X. Liu Guowu Xie

Abstract—With the increasing adoption of cloud computing, a growing number of users outsource their datasets to cloud. To preserve the privacy, the datasets are usually encrypted before outsourcing. However, the common practice of encryption makes the effective utilization of the data difficult. For example, it is difficult to search the given keywords in encrypted datasets. Many schemes are proposed to make encrypted data searchable based on keywords. However, keyword-based search schemes ignore the semantic representation information of users retrieval, and cannot completely meet with users search intention. Therefore, how to design a content-based search scheme and make semantic search more effective and context-aware is a difficult challenge. In this paper, we propose ECSED, a novel semantic search scheme based on the concept hierarchy and the semantic relationship between concepts in the encrypted datasets. ECSED uses two cloud servers. One is used to store the outsourced datasets and return the ranked results to data users. The other one is used to compute the similarity scores between the documents and the query and send the scores to the first server. To further improve the search efficiency, we utilize a tree-based index structure to organize all the document index vectors. We employ the multi-keyword ranked search over encrypted cloud data as our basic frame to propose two secure schemes. The experiment results based on the real world datasets show that the scheme is more efficient than previous schemes. We also prove that our schemes are secure under the known ciphertext model and the known background model.

Index Terms—Searchable encryption, cloud computing, smart semantic search, concept hierarchy.

1 Introduction

Cloud computing is a new but gradual maturity model of enterprise IT infrastructure that provides high quality applications and services [1]. The cloud customers can outsource their local complex data system into the cloud to avoid the overhead of management and local-storage. However, the security of outsourced data cannot be guaranteed, as the Cloud Service Provider (CSP) possesses whole control of the data. So, it is necessary to encrypt data before outsourcing them into cloud to protect the privacy of sensitive data [13]. Li et al. [58] provided a secure privacy-preserving outsourced classification in cloud computing. However, encryption for outsourced data can protect privacy against unauthorized behaviors, it also makes effective data utilization, such as search over encrypted data, a very difficult issue.

In recent years, many researchers have proposed a series of efficient search schemes over encrypted cloud data. The general process of search scheme can be divided into five steps: extracting document features, constructing a searchable index, generating search trapdoor, searching the index based on the trapdoor and returning the search results. These search schemes provide different query capabilities, including single keyword search [2, 3, 4, 5, 6], multi-keyword search [7, 8, 9, 10], fuzzy keyword search [9, 11], similarity search [12], and so on. However, all the existing searchable encryption schemes, which consider keywords as the document feature, do not take the semantic relations between words into consideration, both in the steps of extracting document features and generating search trapdoor. As we all know, the semantic relations between words are diverse [14], such as synonymy and domain correlation. Considering the potentially huge amount of outsourced data documents in the cloud, the search accuracy and search efficiency are influenced negatively if the semantic relations between words are not handled well.

We now give a detailed description of existing problems of the available searchable schemes. Firstly, in the stage of extracting document features, the data owner computes the weight of each word in a document and then selects t words with top-t weights as the feature of the document. In the process shown above, every two words with different spelling are assumed uncorrelated, which is unreasonable. For example, two words "trousers", "pants" are different in the perspective of spelling, but they are semantically similar. It is obvious that the weight of word is influenced if semantic relations between words are ignored and the accuracy of the document features is influenced consequently. Secondly, during generating search trapdoor, the trapdoor is generated only based on the search keywords input by the data user, which is inflexible, because it is impossible to extend the search keywords when the data user cannot express his search intention well. In this case, a useless document can be returned for the data user or the really needed documents are not returned. So, it is important to understand the real search intention of the data user to avoid returning unnecessary documents to improve search

Z. Fu, L. Xia, and X. Sun are with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, 210044 China. E-mail: fzj@nuist.edu.cn, xll20161243602@163.com, sunnudt@163.com.

Alex X. Liu is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824 USA. E-mail: alexliu@cse.msu.edu

G. Xie is with the School of Computer and Engineering, University of California, Riverside, CA 92521 USA. E-mail: xieg@cs.ucr.edu

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

efficiency, as the size of the document set outsourced into the cloud server is potentially huge. Thirdly, a search request usually focuses on a theme, and some search words can be considered to be the attribute of the theme, for example, birthday is an attribute of a person. In existing search schemes, an attribute value is usually treated as a keyword that ignores the relationship with the theme and results in larger keyword dictionary, and then negatively influences the search accuracy and efficiency. Therefore, it is a very important and challenging task to implement semantic search over encrypted data.

In this paper, we propose an efficient searchable encrypted scheme based on concept hierarchy supporting semantic search with two cloud servers. A concept hierarchy tree is constructed based on domain concepts related knowledge of the outsourced dataset. We extend concept hierarchy to include more semantic relations between concepts. With the help of extended concept hierarchy, document features are extracted more precisely and search terms are well extended based on the semantic relations between concepts. For each document, two index vectors are generated, one is used to match concepts in the search request and another one is used to determine whether the value for an attribute is satisfied with the search request. Correspondingly, the search trapdoor for a search request also includes two vectors. The reason why we choose two cloud servers is that two servers can save much time in search. One is used to compute the similarity between the documents vector and the trapdoors vector. Another one is used to rank results and return them to users. Our contributions are summarized as follows:

1) We study the problem of the semantic search based on the concept hierarchy by using two cloud servers. The concept hierarchy is extended to store various semantic relations among concepts and used to extend the search terms. To improve the efficiency and security of the search, the retrieval process is split into two independent procedures.
2) We propose a method to build the document index and search trapdoor based on the concept hierarchy to support semantic search, which filters documents by checking the attribute value and sorts related documents based on the number of matched search terms.
3) The security analysis indicates that our scheme is secure in the threat models. A tree-based searchable index is constructed to improve search efficiency. Experiments on real world datasets show that our schemes are efficient.

Compared with the conference version of the paper[61], we study the problem of semantic search over encrypted data based on concept hierarchy in the basic frame of MRSE. This version also study that how to do the search by using two cloud servers. In addition to these, we do more detailed security analysis of the schemes. Meanwhile, we improve the experiments by doing evaluation of the new schemes.

The rest of the paper is organized as follows. Section 2 introduces the system model, threat model and notations. The detailed description of the concept hierarchy and our secure search scheme are presented in Section3 and Section 4 respectively. In section 5, we introduces the tree-based index structure which is used to improve search efficiency. We analysis the performance of schemes in Section 6. Section 7 summarizes the related work. Finally, the conclusion is given in Section 8.

2 Problem Formulation

2.1 System Model

Compared with the previous version[61], we have an innovation on this version is that we use two cloud servers to search, so we make a new system model. There are four entities in our setting as shown in Fig. 1: the data owner, the data user, the cloud server A and the cloud server B.

Data owner: The data owner encrypts the data held locally and uploads it to the cloud server. In this paper, a concept hierarchy is constructed based on the domain concepts related knowledge of the dataset and two index vectors for each document of the dataset are generated based on the key concepts of the document and the concept hierarchy. Then, the searchable index which is constructed with all the index vectors is sent to the cloud A.

Data users: The authorized data user makes a search request. Then, the trapdoors which related to the keywords are generated. At last, the data user sends the trapdoors to the cloud B.

Cloud Server A: The cloud server A has two functions. One is storing the outsourced dataset. The other one ranks the results from the cloud B and returns the certain encrypted documents that satisfy the search criterion to data users.

Cloud Server B: The cloud server B is used to compute the similarity scores between documents vector and trapdoors vector when it receives the trapdoor. After computing, the cloud B submits these results to the cloud A.

2.2 Threat Model

The previous paper[61] is simply introduced the threat model. Our scheme mainly refers to the dual-servers framework [57] and the MRSE framework [7]. In this version, we consider the cloud server to be semi-honest, which is adopted by most previous works [7],[8],[9], that is to say, who honestly executes the protocol as it is defined and correctly returns the search results, but who may also try to infer private information by analyzing the outsourced dataset, searchable index and query evaluation. And we assume that there is no collusion between two cloud servers. On account of what information the cloud servers learns, we study two threat model [35] as follows.

Known Ciphertext Model The known ciphertext model means that the cloud servers can access the encrypted information which contains the files and indexes outsourced by data owners, but the servers can not understand the plaintext information in the lower layer of the ciphertext.

Known Background Model In this more powerful model, the cloud servers should have much more accessible information compared with known ciphertext model. These information include the encrypted information and the relationship between given search requests (trapdoors) and the data set about the statistical information. As the instance which can be attacked in this situation, the cloud servers can infer/recognize some retrieved keywords by using the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

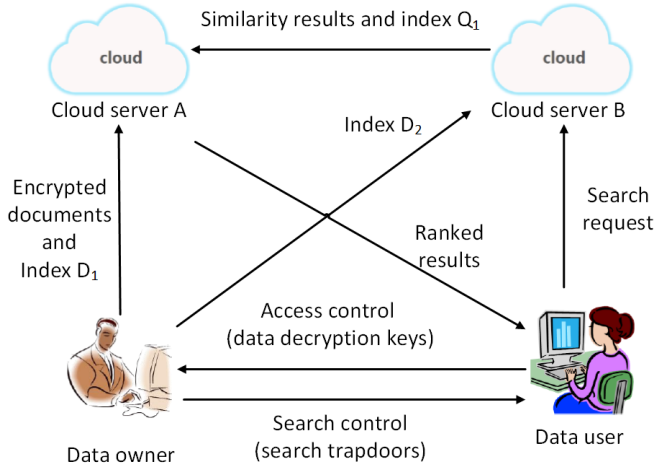

Fig. 1. System model

known trapdoor information and the frequency of documents/keywords.

2.3 Design Goal

We increase the part of design goals to make this paper more clear than previous paper[61]. In order to ensure that our solutions can be implemented accurately and efficiently under the above-mentioned threat models, our schemes must meet two requirements: semantic retrieval based on concept hierarchy and privacy preserving.

The semantic retrieval based on concept hierarchy means that our scheme can compute the similarity scores between the data and the search request and return the ranked results which satisfied the search requests of users.

In this paragraph, we describe privacy preserving in detail. In the search processes under the cloud servers, our schemes must meet the following privacy protection:

1) *Data privacy*. When we provide data documents to users, we also need to ensure the privacy of the document security, which is data privacy. To address this problem, the traditional symmetric cryptography has been proposed. The advantage of this encryption is that we can use a symmetric key encrypted the data documents before outsouring.
2) *Index privacy*. Index privacy is that the cloud servers can not guess the correspondence between the keywords and the encrypted documents through the encrypted index.
3) *Concept privacy*. In this paper, we believe that the concepts and the keywords are linked to a certain degree. Therefore, we need to ensure that the security trapdoor we generated does not reveal the keywords and the query information of users.
4) *Trapdoor unlinkability*. While the cloud servers retrieve documents, it is able to access the generated trapdoors. Therefore, we should guarantee that the randomness of trapdoor generation. At the same time, we need to ensure that the same queries associate with a lot of different trapdoors. In this way, the cloud server can not get relationships which exist in these trapdoors.

2.4 Notations and Preliminaries

The main notations used throughout this paper are shown as follows.

F — the plaintext dataset, denoted by a set of m documents $F = \{F_1, F_2, ..., F_m\}$.

C — the encrypted dataset that outsourced to the cloud server, denoted by $C = \{C_1, C_2, ..., C_m\}$.

KC — the dictionary that contains n key concepts, denoted as $KC = \{c_1, c_2, ..., c_n\}$.

T — the concept hierarchy tree, each node of which corresponds to a concept in KC.

D_{i_1},D_{i_2} — the index vectors of document F_i, where each dimension corresponds to a concept in KC.

$\widetilde{D_{i_1}}$,$\widetilde{D_{i_2}}$ — the encrypted index vectors for F_i.

Q_1,Q_2 — the query vectors for a search request, where each dimension corresponds to a concept in KC.

$\widetilde{Q_1}$,$\widetilde{Q_2}$ — the encrypted query vectors for Q_1 and Q_2.

m — the number of documents in the dataset F.

n — the number of concepts in the concept hierarchy T, also known as the size of T.

Distance comparison function $Comp$ Let $\widetilde{P_1} = E_d(P_1)$ and $\widetilde{P_2} = E_d(P_2)$ be the encrypted form of data points P_1 and P_2. Given $\widetilde{Q} = E_d(Q)$ which is the encrypted form of a query point Q, the function checks whether $(\widetilde{P_1} - \widetilde{P_2}) \cdot \widetilde{Q} > 0$ to determine whether P_1 is nearer to Q than P_2.

We discuss the correctness of the method in brief, which shown as follows. In [35],this method is demonstrated in detail.

As mentioned above, we can deduce

$$\begin{aligned}
&(\widetilde{P_1} - \widetilde{P_2}) \cdot \widetilde{Q} \\
&= [(M_1^T \hat{P_1'}, M_2^T \hat{P_1''}) - (M_1^T \hat{P_2'}, M_2^T \hat{P_2''})]^T \cdot \widetilde{Q} \\
&= [M_1^T(\hat{P_1'} - \hat{P_2'}), M_2^T(\hat{P_1''} - \hat{P_2''})]^T \cdot (M_1^{-1}\hat{Q'}, M_2^{-1}\hat{Q''}) \\
&= (\hat{P_1'} - \hat{P_2'}) \cdot \hat{Q'} + (\hat{P_1''} - \hat{P_2''}) \cdot \hat{Q''} \\
&= (\hat{P_1} - \hat{P_2}) \cdot \hat{Q} \\
&= 0.5r[d^2(P_2, Q) - d^2(P_1, Q)]
\end{aligned}$$

which suggested that if $(\widetilde{P_1} - \widetilde{P_2}) \cdot \widetilde{Q} > 0$, then P_1 is nearer to Q than P_2. Then, we have the comparison function:

$$Comp(\widetilde{P_1}, \widetilde{P_2}, \widetilde{Q}) = \begin{cases} 0, & \text{if } d(P_1, Q) = d(P_2, Q) \\ 1, & \text{if } d(P_1, Q) < d(P_2, Q) \\ -1, & \text{if } d(P_1, Q) > d(P_2, Q) \end{cases} \quad (1)$$

In this paper, we let $P[i]$, the i-th dimension of P, be the dimension needed to be dealt with. And for a query point Q, $Q[i] = \omega$. We use the procedure shown as follows to compare $P[i]$ and ω in encrypted form:

(1) Firstly, we generate two vectors based on Q:

$$\begin{aligned}
Q_a &= (\lambda_1, \lambda_2, ..., \lambda_{i-1}, \omega - h, \lambda_{i+1}, ..., \lambda_n) \\
Q_b &= (\lambda_1, \lambda_2, ..., \lambda_{i-1}, \omega + h, \lambda_{i+1}, ..., \lambda_n),
\end{aligned}$$

where h and $\lambda_j (j = 1, 2, ..., i-1, i+1, ..., n)$ are positive numbers which are randomly generated.

(2) Secondly, we use the function E_d to encrypt Q_a and Q_b, and use function E_q to encrypt P. Then, we can determine the relationship between $P[i]$ and ω using

$$\begin{cases} P[i] = \omega & \text{if } Comp(\widetilde{Q_a}, \widetilde{Q_b}, \widetilde{P}) = 0 \\ P[i] < \omega & \text{if } Comp(\widetilde{Q_a}, \widetilde{Q_b}, \widetilde{P}) = 1 \\ P[i] > \omega & \text{if } Comp(\widetilde{Q_a}, \widetilde{Q_b}, \widetilde{P}) = -1 \end{cases} \quad (2)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

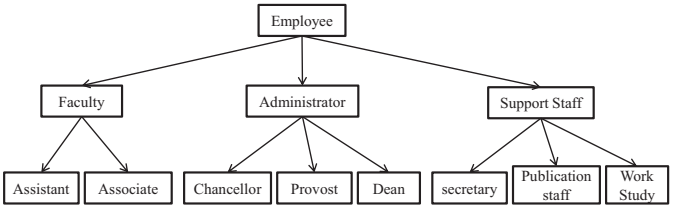

Fig. 2. A concept hierarchy for college employees.

TABLE 1
The semantic relations in the concept hierarchy

semantic relation type	example
synonym	web – network
hypernymy / hyponymy	fruit – apple
meronym / holonym	forest – tree
host – attribute	people – name
attribute – value	color – red

3 CONCEPT HIERARCHY

A concept hierarchy is an organized concept set using hierarchical method. In the hierarchy, the concepts at lower levels contain more specific meanings than those at higher levels. To make it easy to understand, we give an example in Fig. 2 to illustrate the concept hierarchy. As mentioned before, the concept hierarchy is constructed based on the semantic relationships between concepts, that is to say, the concept hierarchy contains semantic information between concepts.

3.1 Extended Concept Hierarchy

As we know, the text file is unstructured, but the language organizing the text file exists semantic relation, which can be regarded as an implicit structure. In this paper, we use an extended concept hierarchy to denote the semantic relationship between concepts. The semantic relations contained in our extended concept hierarchy are shown in Table I.

The main difference between our extended concept hierarchy and concept hierarchy is that, the concept can possess attribute, which can be assigned different values. Some concepts can possess some attributes that are also concepts. We take concept "doctor" as an example. A "doctor" can have attributes like "name", "gender", and so on. The attribute information can make the search result more accurate, as it makes the search request more specific. And as the concept and attribute are organized via the concept hierarchy, the semantic relationship is preserved. We take Fig. 3 as an example to illustrate the extended concept hierarchy. As shown in Fig. 3, the attribute "color" of concept "jersey" has a value "red". Note that the concept acted as attribute has not relation with other concepts. The hypernymy/hyponymy relation and synonym relation can also be seen in Fig. 3.

This paragraph shows that the process of generating the extended concept hierarchy. At first, we generate the concept hierarchy based on the domain information of the outsourced dataset. And then we deal with the dataset to extend the concept hierarchy. The domain concept hierarchy can be obtained by some existing tool, such as WordNet [14],

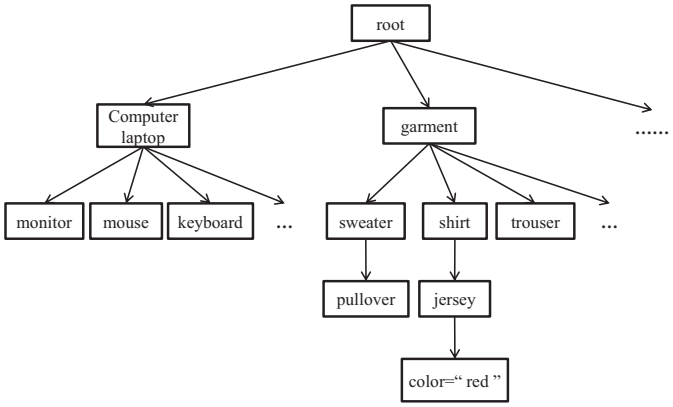

Fig. 3. An example of our extended concept hierarchy.

the hyponymy of which could be regarded as relationship of superclass and subclass, or some existing concept hierarchy, such as web directory ODP(open directory project) [34]. As we all know the content of a document usually focus on a subject that can be denoted by concepts and relations of concepts. For example, the sentence "A paper about economy is published in the newspaper on March 5, 2014" is the subject of a document, which can be denoted by concepts and relations among them. There are many subject extraction techniques with which the subjects of files are obtained. And then the concepts and its relations are generated to extend the concept hierarchy. As WordNet includes all the semantic relationships that adopted in the extended concept hierarchy, we utilize it to construct our extended concept hierarchy. The values of a certain attribute concept are determined by the outsourced dataset.

3.2 Semantic Similarity between Concepts

We now give a detailed description of how to evaluate the semantic similarity between two concepts is given. The similarity between two concepts will be used in the stage of query to extend search terms.

After the concept hierarchy is built, the semantic similarity between two concepts can be calculated. The similarity between two concepts is calculated based on the distance of them in the concept hierarchy. Given two concepts c_1 and c_2, we denote the distance between them by $dis(c_1, c_2)$. The similarity between them can be calculated as $sim(c_1, c_2) = 1 - dis(c_1, c_2)$. The issue of calculating the distance between two concepts has been studied by some previous work [31, 32, 33]. We make some modifications on their original thought to fit our requirement. Each node z in the concept hierarchy owns a value, denoted by $Score(z)$, which can be calculated according to Eqn.(1).

$$Score(z) = \frac{1/2}{k^{d(z)}} \tag{3}$$

where k is a factor which is defined as 2 in this paper, and $d(z)$ is the depth of node z in the concept hierarchy. Note that for the root node of the concept hierarchy tree, we define $d(root) = 0$.

For two concepts c_1 and c_2, we let concept cp be the closest common parent of them. Then, the distance between c_1 and c_2 can be obtained by Eqn.(2).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

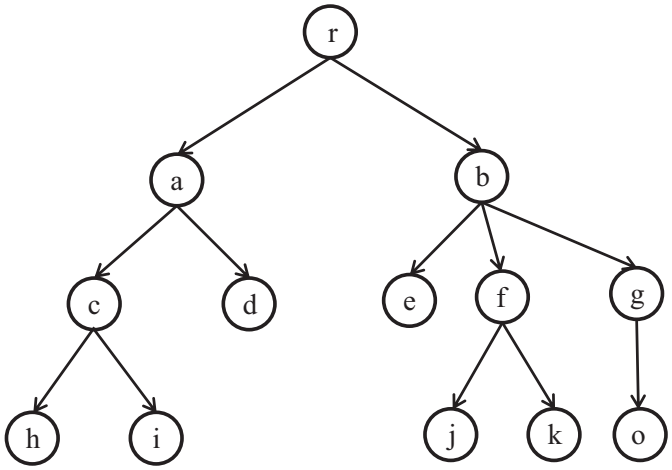

Fig. 4. A concept hierarchy T.

$$\begin{cases} dis(c_1, c_2) = dis(c_1, cp) + dis(c_2, cp) \\ dis(c, cp) = Score(cp) - Score(c) \end{cases} \quad (4)$$

This evaluation method for semantic similarity provides the feature that the differences between two concepts in lower level are smaller than those between concepts in upper level. And the method also supports our requirement that the similarity between "father" and "son" should be greater than that between "brothers".

4 Secure Search Scheme

4.1 Generating Document Index Vector

As we introduce "attribute-value" relation in the hierarchy, two index vectors should be generated for each document in the dataset, one vector is used to match concepts in the search request and another one is used to determine whether the value for an attribute is satisfied with the search request. The process of generating these two n-dimension index vectors based on the extended concept hierarchy is shown as follows. For a document F, we denote its two index vectors by D_1 and D_2. Each dimension of D_1, denoted by $D_1[i]$, corresponds to a node (stores concept c_i) in the hierarchy. If F contains the concept c_i, then $D_1[i] = 1$, otherwise $D_1[i] = 0$. Similarity, each dimension of D_2, denoted by $D_2[i]$, corresponds to a node (stores concept c_i) in the hierarchy. If F contains c_i and c_i has a value in F, denoted by $val(c_i, F)$, then $D_2[i]$ is:

$$D_2[i] = \begin{cases} H(val(c_i, F)) & \text{if } val(c_i, F) \text{ is string} \\ val(c_i, F) & \text{if } val(c_i, F) \text{ is number,} \end{cases} \quad (5)$$

where H is a hash function denoted by: $H : \{0,1\}^* \rightarrow \{0,1\}^k$, otherwise, $D_2[i] = 0$. If c_i is not a attribute concept, then $D_2[i] = \alpha_i$, where α_i is a random number. We take the concept hierarchy T in Fig. 4 as an example to illustrate the process. And in T, the concepts j, k, o are attribute nodes. Suppose that a document F contains three concepts b, f, j and j is an attribute concept with value "1994". The index vectors for F are shown in Fig. 5.

nodes	a	b	c	d	e	f	g	h	i	j	k	o
D_1	0	1	0	0	0	1	0	0	0	1	0	0
D_2	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	1994	0	0

Fig. 5. Index vectors for document F.

nodes	a	b	c	d	e	f	g	h	i	j	k	o
Q_1	0	1	0	0	1	1	0	0	0	1	0	0
Q_2	0	0	0	0	0	0	0	0	0	1990	0	0

Fig. 6. Search trapdoor for search request Q.

4.2 Generating Trapdoor

A search request consists of several concepts. Once receiving a search request, the procedure calculates the semantic similarity between each search concept and its candidate concepts in the extended concept hierarchy using function defined in Section 3.1. For each search concept, the candidate concepts are its "brother", "father", and "direct child" nodes in the concept hierarchy. We let γ be a parameter to determine whether a candidate concept deserve to be added to search requests. To be specific, for a search concept c_1 and its candidate concept c_2, if $sim(c_1, c_2) \geq \gamma$, then we add c_2 to the search request. Finally, the extended search request is generated. Note that we do not try to deal with attribute concepts in the concept extending process above.

For a search request containing several concepts, two n-dimension vectors are also generated, one is used to store the information about concepts in the search request and another one is used to store the search restriction on attribute. For a search request Q, we denote its two search vectors by Q_1 and Q_2. The method of generating the value of each dimension of Q_1 is similar to that for D_1, that is, if Q contains c_i, then $Q_1[i] = 1$, otherwise $Q_1[i] = 0$. For the extended search concepts, suppose c_j is a concept extended from search concept c_i and $sim(c_i, c_j) = \mu$, then $Q_1[j] = \mu$. For vector Q_2, if there exist restriction on the value of attribute concept c_i and the constraint value is λ, then $Q_2[i] = \lambda$, otherwise $Q_2[i] = 0$. We also take T in Fig. 4 as an example to illustrate the process. Assume that a search request Q contains concepts e, b, f, j after concept extending process, where j is a attribute concept whose value should satisfy $val(j) > 1990$. The search trapdoor Q_1 and Q_2 for Q is shown in Fig. 6.

Given the index vectors of a document F and the search trapdoor of a search request Q, the search procedure is conducted as follows. Firstly, the procedure checks whether the document satisfies search restrictions included in search request using vectors D_2 and Q_2. Secondly, if D_2 satisfies Q_2, then the procedure computes $D_1 \cdot Q_1$ to obtain the similarity score of the document to the search request, where the value of $D_1 \cdot Q_1$ indicates the number of matched concepts between F and Q. At last, all the related documents are sorted based on their similarity scores and the top-k related documents are returned to the user, where k is a parameter received from the user.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

4.3 $ECSED$-1 Scheme: Secure Scheme in Known Ciphertext Model

In this paper, we replace the previous framework ASPE[61] with the framework MRSE to improve the efficiency and security. We use MRSE [7] as our basic framework. To improve the efficiency and security, we split the retrieval process into two parts and carry out them in two servers, respectively. Meanwhile, we extend the dimension of the vectors to $(n+2)$ for reducing the possibility that the cloud servers can infer the relationship between the trapdoor and the index. Thus, we propose the secure scheme $ECSED$-1 under the known ciphertext model. The specific algorithms are as follows.

Setup: The security parameter λ as input, and then this algorithm generates a public parameter P.

SKeyGen: The data owner randomly generates a secret key SK, which is in the form of a 3-*tuple* as $\{S, M_1, M_2\}$.

BuildIndex: For each document F in the dataset, the data owner generates two index vector D_1 and D_2 using the procedure in Section 4.1. The differences are that we give a random value ε_i to the $(n+1)$th entry in the $\vec{D_1}$ and the $(n+2)$th entry in the $\vec{D_1}$ is set to 1. We express the $\vec{D_1}$ as (D_1,ε_i, 1). Finally, the subindex $D_1 = \{{M_1}^T\vec{D_1}', {M_2}^T\vec{D_1}''\}$ for the document has been built. We define the encrypted form of D_1 and D_2 as $\widetilde{D_1} = E_d(D_1)$ and $\widetilde{D_2} = E_q(D_2)$. At last, we send $\widetilde{D_1}$ and $\widetilde{D_2}$ to the cloud server A and B respectively. Note that, the dataset is also uploaded to the cloud server A by the data owner.

Trapdoor: For a search request Q, the data user generates two vectors Q_1 and Q_2 by using the procedure in Section 4.2. We use MRSE to encrypt the vector Q_1. So, we can get $\vec{Q_1}$, which equals to $\{{M_1}^{-1}\vec{Q_1}', {M_2}^{-1}\vec{Q_1}''\}$. Suppose that i is a dimension of Q_2 that exists attribute value restriction and the restriction condition is "$val(c_i) \quad op \quad Q_2[i]$", where $op \in \{>, \geq, <, \leq, =\}$. Then, for each restriction condition, two vectors are built based on Q_2:

$$\begin{aligned} Q_{2a} &= (\lambda_1, \lambda_2, ..., \lambda_{i-1}, \omega - h, \lambda_{i+1}, ..., \lambda_n) \\ Q_{2b} &= (\lambda_1, \lambda_2, ..., \lambda_{i-1}, \omega + h, \lambda_{i+1}, ..., \lambda_n), \end{aligned}$$

where $Q_2[i] = \omega$ and h and $\lambda_j (j = 1, 2, ..., i-1, i+1, ..., n)$ are randomly generated positive numbers. Then Q_{2a} and Q_{2b} are encrypted by using MRSE in the form of $\widetilde{Q_{2a}}$ and $\widetilde{Q_{2b}}$. At the end, the Q_1 and each pair of ($\widetilde{Q_{2a}}$, $\widetilde{Q_{2b}}$, *op*) are sent to the cloud server B as the search trapdoor.

BTest: This process has been done under the cloud server B. In the BTest, the procedure checks whether a document F should be returned in two steps : 1) for each pair of ($\widetilde{Q_{2a}}$, $\widetilde{Q_{2b}}$, *op*), Eqn.(6) is used to determine whether D_2 meets the restriction condition; 2) if D_2 satisfies all restriction conditions, then we can get the similarity between the query and document F with the given t for the query vector and the given ε_i for the data vector.

$$Comp(\widetilde{Q_{2a}}, \widetilde{Q_{2b}}, \widetilde{D_2}) = \begin{cases} 0, & \text{if } op = \text{`} = \text{'} \\ 1, & \text{if } op = \text{`} < \text{'} \\ -1, & \text{if } op = \text{`} > \text{'} \end{cases} \tag{6}$$

$$\begin{aligned} I_i \cdot T &= \{{M_1}^T D_1, {M_2}^T D_2\} \cdot \{{M_1}^{-1} Q_1, {M_2}^{-1} Q_2\} \\ &= r(D \cdot Q + h_i) + t \end{aligned} \tag{7}$$

ATest: We execute the ATest under the cloud server A. The main function of this algorithm is to sort the results which returned from BTest, and then return the first k files which meet the user requirements to data users.

4.4 $ECSED$-2 Scheme: Secure Scheme in Known Background Model

Compared with the conference version[61], this paper make new schemes for new threat models. For the known background model, the security of the above scheme is not high enough because the cloud server understand some information of the background relationships between the trapdoors and the specific keywords, it is possible to infer the specific keyword by the hidden information of the trapdoors. As a instance, the cloud servers can guess some high frequency keywords, through the known background information and the documents frequency. Therefore, we propose a more secure solution $ECSED$-2 to resist the known background attack.

Since the value of a particular random variable ε_i in the file is fixed, the privacy of the above scheme $ECSED$-1 can be let out. To solve this problem, we need to add more dummy keywords to the data vector $\vec{D_1}$. That is, the original $(n+2)$-dimensional vector is extended to the vector of $(n+U+1)$ dimensions. In this way, the security of our solution will be significantly improved. The improved scheme is as follows:

Setup: The security parameter λ as input, and then this algorithm generates a public parameter P.

SKeyGen: This algorithm is almost the same as the previous scheme $ECSED$-1. The only difference is that the vector S of the secret key SK is expanded from the original $(n+2)$-dimension to $(n+U+1)$-dimension and the two matrices also be changed into $(n+U+1) \times (n+U+1)$ dimensions.

BuildIndex: When we extend the vector into $(n+U+1)$ dimensions, we give the $(n+j+1)$th entry in $\vec{D_1}$ a number randomly ,where $j \in [1, U]$. After it, the procedures like spilt the vector are same as the $ECSED$-1.

Trapdoor: The query is generated as above by the data users. And we select V dummy keywords from the U dummy keywords randomly to assigned into the value of 1.

BTest: In this algorithm, the cloud server computes the similarity between the data vector D_1 and the query vector Q_1, and the result of computation contains one of the V dummy keywords and it can be represented as ($D_1 \cdot Q_1$ + $\Sigma\varepsilon_i^v$)+t_i.

ATest: This algorithm is as same as the previous scheme $ECSED$-1.

Based on this approach, the cloud server is more difficult to obtain useful information from the index and trapdoor. We refer to the relationship between U and V as well as the analysis from [7]. We take into account the accuracy and efficiency, selecting the appropriate U and V to participate in the calculation.

4.5 Security Analysis

Compared with the previous version[61], due to the modification of the framework in this paper, we focus on the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

security analysis of our scheme based on MRSE framework [7] and the dual-servers framework [67]. We all know that MRSE is adapted from the secure kNN algorithm [35]. Therefore, we only need to prove the kNN algorithm and MRSE are sufficiently secure to demonstrate that our $ECSED$-1 and $ECSED$-2 are secure. We will prove that our scheme is secure enough under the known background model. For the known ciphertext model [4, 43, 45, 46], we will directly employ the analysis of MRSE [7]about scale analysis attack and adopt its results.

Size Pattern We denote D for the plaintext document set which has n documents. $\{n, |Q_1|, ..., |Q_m|\}$ is the collection for the size pattern of query Q with the length of m ,where the length of query Q expressed as $|Q|$.

Access Pattern The plaintext document set containing n documents has been made as D and we named the set of generated index as I. The collection as follow:$\{I(Q_1), ..., I(Q_m)\}$ express the access pattern of query Q based on size pattern. The set of index which is linked to Q has been expressed as $I(Q)$ in the collection.

Search Pattern We make a plaintext document set containing n documents as D. The search pattern consists of a $n \times q$ matrix M. In the matrix, the value of each dot means that whether the specific query Q in the size pattern exists in the file.

Known Ciphertext Model We denote the Ω = (*Setup, skeyGen, Buildindex, Trapdoor, BTest, ATest*) to indicate the security parameter for our schemes of $ECSED$-1 and $ECSED$-2. We summary the secure experiment (SE) under the known ciphertext model as follows:

1) The challenger will get two documents with same size which are submitted by adversary.
2) The challenger runs Setup and skeyGen to get a secret key $\{M, S\}$.
3) In a random position of the index vector, a random index which is filling by 0 or 1 is generated randomly by the challenger. Then, the challenger sends the encrypted random vector to the adversary.
4) The adversary achieves the random index by doing the step 3.
5) When the results of Step 3 and Step 4 are identical, we will mark the result of the experiment as 1.

For all probabilistic polynomial-time adversaries, as long as the probability of the above secure experiment is not greater than $0.5 + r$, we assume that our experimental $ECSED - 1$ and $ECSED - 2$ under the ciphertext model are secure. The r in the formula is a negligible random number.

$$Pr(SE = 1) \leq 0.5 + r \tag{8}$$

Proof: From the origin of information which is used to distinguish the authenticity by the adversary, we obtain a formula as follow:

$$Pr(SE = 1) = 0.5 + D(M, S) + D(I) + D(ED) \tag{9}$$

In the formula, $D(M, S)$ represents the advantage of the opponent can distinguish M and S from two random matrices and two random strings, as same as $D(M, S)$, $D(I)$ means the index and $D(ED)$ express the encrypted files.

But for $D(M, S)$, it is very difficult to distinguish M and S from two random matrices and two random strings, and the possibility is almost negligible. The reason is that both the matrix and the key are composed of random and uniform repeate by $\{0, 1\}$.

We use a secure KNN algorithm to encrypt the generated index. In [35], the security of the kNN algorithm to resisting the known plaintext attack has been well proved. That is to say, we can get such a conclusion: In our schemes, the probability of the adversary can guess the real index is very small and negligible.

Next, in order to prove security, we just need to prove that the probability of occurrence of $D(I)$ is small enough to be ignored. The existing encryption algorithms have certain semantic security to ensure that we encrypted documents in the known ciphertext model are secure enough. Therefore, the probability of $D(I)$ can be ignored.

Based on all of the above analysis, we know that $D(M, S) + D(I) + D(ED)$ is small enough to be negligible. Thus, formula (8) is established to show that our schemes are secure sufficiently under the ciphertext model

Dual-servers framework In this paper, we assume that both servers are "honest but curious", and there is no collusion between them. These two cloud servers mainly resist semantic-security against chosen keyword attack and indistinguishability against keyword guessing attack. Based on the definition of the security model in refference [67], we give the definition of the security of the scheme. More details can be seen in [67].

$Definition$ We say that a Dual-servers framework is secure if for any polynomial time attacker A_i $(i = 1, ...5)$, we have that $Adv_{B,A_1}^{SS-CKA}(\lambda)$, $Adv_{B,A_2}^{SS-CKA}(\lambda)$, $Adv_{A,A_3}^{IND-KGA}(\lambda)$, $Adv_{B,A_4}^{IND-KGA}(\lambda)$ and $Adv_{B,A_5}^{IND-KGA-II}(\lambda)$ are all negligible functions of the security parameter λ.

5 Search Efficiency Improvement

In our secure search scheme described in Section 4, it is necessary to check every document in the outsourced dataset for a search request, which is inefficient if the size of the outsourced dataset is huge. We propose a method to improve the search efficiency so as to make the search scheme efficient for large-scale dataset. We build a k-d tree to organize all the document index vectors so as to cluster documents with similar attribute values in the same leaf in a k-d tree.

A k-d tree (k-dimension tree) is a binary tree that is used to index multi-dimension data points. Each node of a k-d tree stores a k-dimension data point dp and one dimension $x \in \{1, 2, ..., k\}$ used for splitting data space and two pointers pointing to its right and left subtrees. For every data point l_dp in the left subtree of dp, $l\_dp[x] \leq dp[x]$, and for every data point r_dp in the right subtree of dp, $r\_dp[x] > dp[x]$. A k-d tree is built from the root node. Given a set of data points, suppose that the i-th dimension is used for splitting in the root node, then all the values of i-th dimension are sorted and the data point having the median value is selected to split data points into two data spaces.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

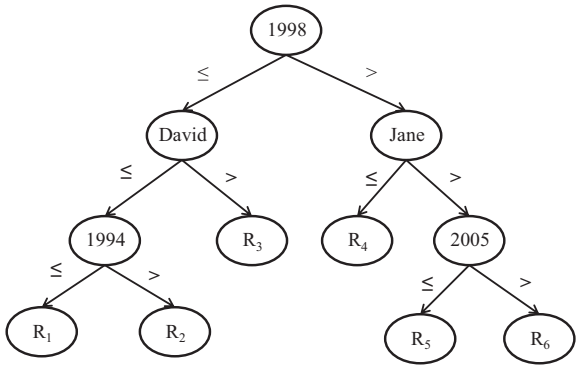

Fig. 7. Index vectors for document F.

And the splitting process is recursively implemented in the left and right subtrees of root node. Usually, the dimension used to split in a node is decided in a round-robin way. To be specific, the splitting dimension for a node at level lp is the i-th dimension, where $i = lp \ mod \ (k) + 1$.

In this paper, all attribute concepts in the concept hierarchy are used to build the k-d tree. The value of one attribute concept can be numerical or categorical. The numerical values can be used directly, but categorical values should be mapped into integers. Now, we introduce some mapping methods. The previous work [37] provides a method for mapping categorical values, which assigns each value a distinct integer based on its pre-order traversal in a hierarchy that used to organize these values. A categorical value can be mapped to the number of times that the value occurred in the dataset when such a hierarchy is unavailable. Furthermore, categorical values can also be organized by their initials. For each document F of the outsourced dataset, two index vectors D_1 and D_2 are generated and D_2 stores the attribute value related information (see details in Section 4). So, in our k-d tree, we use all the D_2 vectors to act as splitting data points, that is to say, there is a D_2 vector in each internal node. And all the D_1 vectors only exist in leaf nodes. In the search stage, the procedure searches from the root node and determine continue to search in the left subtree or right subtree by evaluating the vector D_2 in root node with the query vector Q_2. Once arriving at a leaf node, the procedure computes the similarity value between D_1 and Q_1. Finally, the ordered search results are returned to the data user.

We give a simple example to illustrate the k-d tree. Suppose there are two attributes "year" denoting when the paper is published and "author" denoting the author of the paper, the k-d tree is built based on the two attributes, where k=2. In Fig. 7, the k-d tree is built using the building process stated above, where the dataset is divided into six parts. With the k-d tree, a small subset of leaf nodes is evaluated for a search request. Therefore, the search efficiency is improved.

6 Performance Analysis

Because we modify the framework of ASPE[61], so we make new experiments. In this section, we realize the search system using C# language to estimate the overall performance of the proposed scheme. The implementation platform is Windows7 server with Core2 CPU 2.93GHz. In the experiment, we choose a real-world dataset: Enron Email Dataset [36] which is publicly available to build the outsourced dataset. We choose 10,000 files from Enron Email Dataset as the outsourced dataset. Then we build the concept hierarchy based on the dataset with the help of WordNet. Finally, we construct the concept hierarchy with 10,351 concepts.

6.1 Precision

Our experiments have carried out a precision analysis of our scheme. As described in Part IV, we add virtual keywords to each vector to improve our privacy. However, these virtual keywords affect our documents similarity scores and influence our search results. That is to say, the cloud server retrieves the top-k files to the users based on the similarity score. However, these may not contain the real related files because of their similarity scores being reduced or other documents similarity scores being increased by the addition of virtual keywords. Therefore, we will introduce the method of balancing privacy and accuracy in [7] to our schemes. According to the analysis of this strategy in [7], we use its balance parameters to meet the user's privacy and accuracy requirements.

6.2 Efficiency

In this section, we conduct a comprehensive analysis of the experiment results from three aspects: the establishment of indexes, the generation of trapdoor, and the query process. Through the analysis of the experiment results, we prove the schemes which implement the semantic search by using two cloud servers under the encrypted cloud are much more efficient than the previous schemes (MRSE).

6.2.1 Index Construction

For building index, we construct a searchable subindex for each document F_i in the document set F. We can divide the process into three steps. At first, a set of keyword set should be extracted from the document set F. Then, according to the keyword set, data vectors are generated. The last step is encrypting these data vectors by MRSE. Throughout the process, the time of mapping and encryption is the most cost. And the factor that can affect this time directly is the dimension of the data vector. However, this is just a one-time operation based on the number of documents, which is acceptable. As shown section 4.3, the main calculation when generating the index is the splitting process and two multiplications of a $(n+2) \times (n+2)$ matrix and a $(n+2)$-dimension data vector, which are all influenced directly by the size of dictionary. According to this, we can learn that the time complexity of $ECSED$-1 is $O(mn^2)$ in the theory. Due to the dimensionality of matrices in the $ECSED$-2 is $(n+U+1) \times (n+U+1)$, the time complexity is $O(m(n+U)^2)$, which is little bigger than $ECSED$-1's.

In section V, we introduce a tree-based index structure to optimize the search efficiency. The tree construction procedure should execute sort operation in each internal node and the time complexity is based on the number of index vectors. For a k-d tree built for m files, the computation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

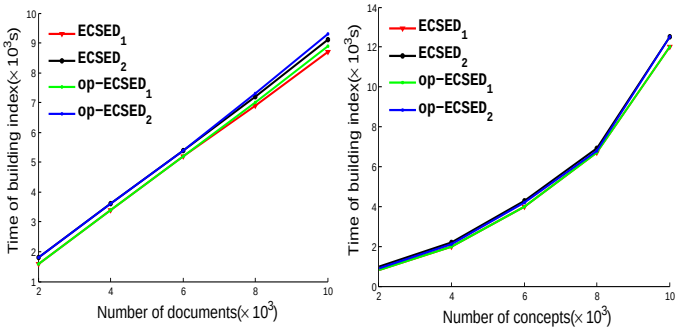

Fig. 8. Time of index construction. (a) For the different number of documents in the dataset with the same number of concepts, n=4000. (b) For the different number of concepts in the concept hierarchy with the same dataset, m=1000.

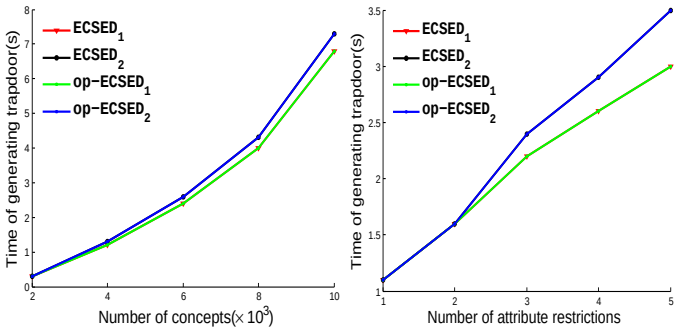

Fig. 9. Time of trapdoor generation. (a) For the different number of concepts in the concept hierarchy with the same number of attribute restriction. (b) For the different number of attribute restrictions with the same number of concepts, n=4000.

complexity of building the tree is $O(k \cdot m \cdot \log m)$. We evaluate the time of index construction for basic search scheme and optimization scheme respectively.

In Fig. 8, we compare the time of index construction between the basic schemes($ECSED$-1, $ECSED$-2)and the optimization schemes(op-$ECSED$-1, op-$ECSED$-2). From the Fig. 8(a), we can see that when the number of concepts is same, the time of index construction is linear with the number of documents in all the schemes. And we can see the time of $ECSED$-2 and its optimization scheme is higher than the others, because of its high dimensions. Due to the extra time cost of tree index, the optimization schemes use little more time than the basic schemes. Fig. 8(b) shows the relationship between index construction time and the size of concept hierarchy for basic schemes and optimization schemes within same dataset. The construction time for these schemes is almost proportional to the size of concept hierarchy. Note that for optimization schemes, as the number of index vectors (the size of dataset) is constant, the time for tree construction is almost constant for different size of concept hierarchy. So with the increasing of the number of concepts, the difference of the index construction time for basic schemes and optimization schemes is almost constant, which can be seen in Fig. 8 (b).

6.2.2 Trapdoor Generation

The time taken to generate the trapdoor mainly depends on the number of nodes in the concept hierarchy. Similar to the index construction, the generation of each trapdoor also includes two matrix multiplications and a split query vector with Q_1 and Q_2. Q_2 is converted into several couple of $(n+2)$-dimension vectors according to attribute restriction conditions the dimension of the matrix and the vector, which is associated with the dictionary, is different between our schemes. Thus, the time cost of generating trapdoor in the basic schemes with complexity $O((n)^2)$ is a little smaller than that in the optimization schemes with the complexity of $O((n+U)^2)$. From the Fig. 9(a), we know the relationship between query generation time and the size of concept hierarchy. With the growth of the size of concept hierarchy, the query generation time grows linearly. Fig. 9(b) indicates the relationship between query generation time and the number of attribute restrictions which determines the number of index vectors that needed to be encrypted. From Fig. 9(b), we know that the query generation time grows linearly along with the growth of the number of attribute restrictions.

6.2.3 Query

For each query Q, the search procedure has three steps. At first,the cloud server B checks whether the index vector D_2 for document F satisfies the search vector Q_2. Then, the operation of computing the similarity score between F and Q is done by the cloud server B when D_2 satisfied Q_2. The last step of the query process is that the cloud server A ranks all the similarity scores of relevant documents and return the documents which meet the requirements to the data user. Thus, the search time is mainly affected by two factors: the size of dataset, and the size of concept hierarchy. The time complexity of the scheme $ECSED$-1 and op-$ECSED$-1 is $O(mn)$, compared to the above schemes, $ECSED$-2 and op-$ECSED$-2 come to with a little bit of growth. Fig. 10(a) shows that with the same size of concept hierarchy, the search time is almost linear to the size of dataset for $ECSED$-1, $ECSED$-2 and their optimizations. Fig. 10(b) indicates the relationship between search time and the size of concept hierarchy within same dataset, where the search time is proportional to the number of concepts. Fig. 10 indicates that the search efficiency is better improved with the optimization scheme.

6.2.4 comparison

In the end, we compare our schemes with MRSE schemes[7], where MRSE uses keyword as document feature. From Fig. 11 and Fig. 12, we can see that the time of generating index and trapdoor in MRSE is faster than our schemes. But our schemes have higher precision in searching. Moreover, in Fig. 13, it can be know that search efficiency of our optimization schemes is better than MRSE schemes.

7 Related Work

7.1 Searchable Encryption based on keyword

Searchable encryption schemes usually generate a searchable index based on the keyword dictionary, which is extracted from the outsourced dataset, and upload the encrypted index together with encrypted dataset to the cloud

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

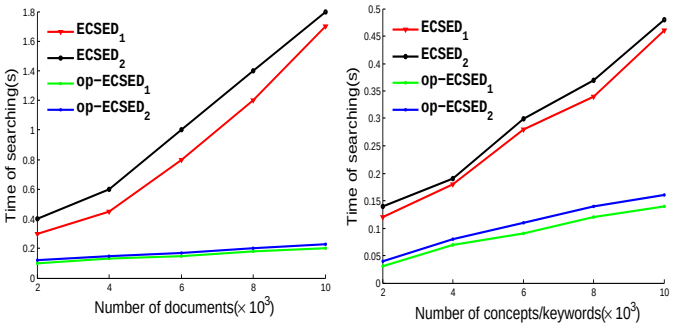

Fig. 10. Search time. (a) For the different number of documents in the dataset with the same number of concepts/keywords, n=4000. (b) For the different number of concepts/keywords with the same dataset, m=1000.

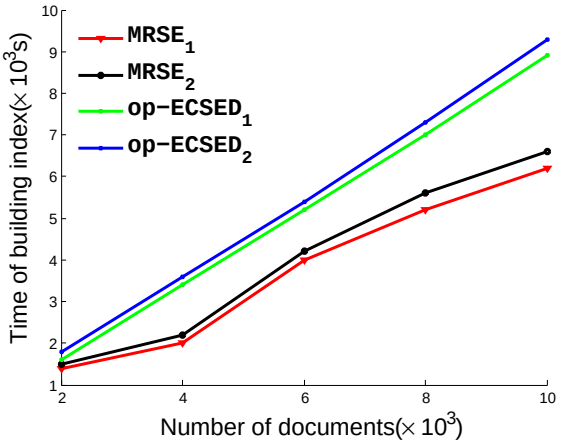

Fig. 11. Time of index construction.

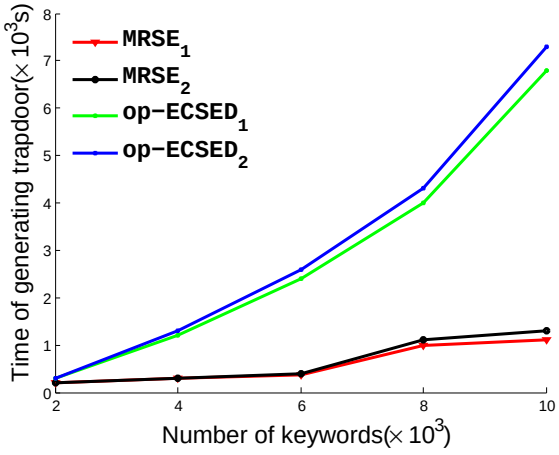

Fig. 12. Time of trapdoor generation.

server. With the trapdoor generated in the search stage, the server can search the searchable index and return related documents. Traditional searchable encryption schemes [3, 4, 5, 6, 15] only support single keyword search and take inverted index as its index structure. In order to improve the functionality and usability of the search system, some works are focused on fuzzy keyword search, similarity search and

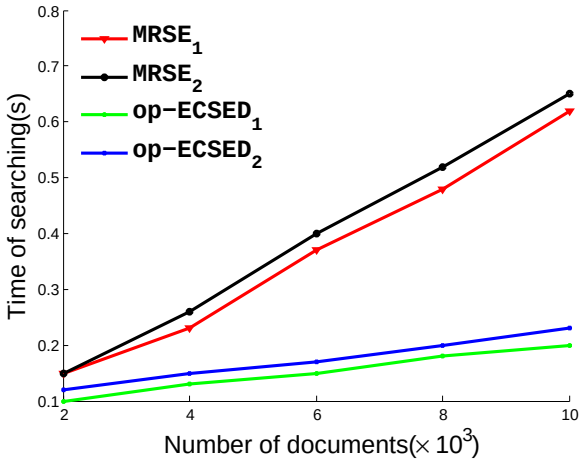

Fig. 13. Search time.

ranked search. Scheme in [11] uses edit distance to extend keyword dictionary to provide fuzzy keyword search. [12] solves the issue of similarity search, which introduces a tire tree to enhance search efficiency. By utilizing keyword weight and orderpreserving encryption technique, schemes [2, 5, 6] can rank search results and return most relevant documents.

As multi-keyword search can provide more accurate search results, some works are concentrated on the issue of multikeyword encryption search [7, 8, 9, 10, 59] in the symmetric setting. Cao et al. [7] proposed a searchable encryption scheme which supports multi-keyword ranked search, where coordinate matching is used to conduct result ranking. The scheme does not take the keyword weight within document into consideration, which makes the search result not accurate enough. Sun et al. [8] proposed a secure multikeyword search scheme supporting similarity-based ranking, which adopts vector space model to build its searchable index and builds a MDB-tree [16] to enhance the search efficiency. Chen et al. [54] presented new algorithms for secure outsourcing of modular exponentiations. This methods can solve the problem that there is no single trusted user. Reference [60] proposed an efficient encrypted keyword search scheme for multi-user data sharing. This scheme balance the security and the search cost. Many works [17, 18] have been done in the public key setting, which support conjunctive keyword search, subset search and range queries. However, schemes in the public setting usually need to sustain more computational burden, such as the bilinear map operation in [17, 19, 20, 21] focus on predicate encryption which supports both conjunctive and disjunctive search, however, without supporting result ranking. References [49, 51, 56] presented data auditing schemes for cloud computing. To improve efficiency, references [50, 52] used several new algorithms. Li et al. [62] proposed a secure attribute-based data sharing scheme.

7.2 Semantic search

Semantic search [22, 23, 24] becomes increasingly important and more and more researchers engaged in the field, as traditional keyword based search scheme cannot exploit

the hidden meanings of terms and the semantic similarity between terms. By utilizing some semantic tools, such as linguistic ontology, concept hierarchy [25, 26, 27], the semantic search scheme can improve both precision and recall. The concept hierarchy, a semantic tool used for organizing concepts, is mainly constructed to indicate the relationships between concepts. The most important usage of concept hierarchy is to distinguish meanings for classification [28, 29] or exploit semantic similarities [30]. Some related works are focused on the issue of semantic distance [31, 32, 33] based on concept hierarchy. The basic idea to define the semantic distance between two concepts is based on the number of arcs in the shortest paths of two concepts in the concept hierarchy. References [47, 48] also presented semantic search methods for encrypted cloud data.

The search keywords always carry semantic information, so we can use this information to do semantic search. Fu et al. [55] proposed the central keyword extension semantic search which improve the relevance of query results. However, in the aspect of semantic search, the scheme based on the concept hierarchy in this paper is better than the schemes based on the extended central keyword. Fu et al. [44, 53] presented search methods based on concept graph, which are initial and intuitive solutions to solve the problem of semantic searchable encryption. The schemes are less efficient than the scheme in this paper, because the construction of concept graph is more complex. Compared with these schemes, our proposed methods keep the balance between the efficiency and the semantics.

8 Conclusions

In this paper, to address the problem of semantic retrieval, we propose effective schemes based on concept hierarchy. Our solutions use two cloud servers for encrypted retrieval and make contributions both on search accuracy and efficiency. To improve accuracy, we extend the concept hierarchy to expand the search conditions. In addition, a tree-based index structure is constructed to organize all the document index vectors, which are built based on the concept hierarchy for the aspect of search efficiency. The security analysis shows that the proposed scheme is secure in the threat models. Experiments on real world dataset illustrate that our scheme is efficient.

Acknowledgments

This work is supported by the NSFC (61772283, 61672294, U1536206, 61502242, U1405254, 61602253), BK20150925, R2017L05, PAPD fund, Project funded by China Postdoctoral Science Foundation, Major Program of the National Social Science Fund of China (17ZDA092), Qing Lan Project, and Meteorology Soft Sciences Project.

References

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2009.

[2] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE TPDS*, vol. 23, no. 8, pp. 1467–1479, 2012.

[3] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. of S&P*, 2000.

[4] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," In *Proc. of ACM CCS*, 2006, pp. 79–88.

[5] A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A. L.Varna, S. He, M. Wu, and D. W. Oard, "Confidentiality-preserving rank-ordered search," in *Proc. of the 2007 ACM Workshop on Storage Security and Survivability*, 2007, pp. 7–12.

[6] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k retrieval from a confidential index," in *Proc. of EDBT*, 2009, pp. 439–449.

[7] N. Cao, C. Wang, and M. Li, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *Parallel and Distributed Systems,IEEE Transactions on*, vol.25,no.1, pp.222-233,2014.

[8] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, YT. Hou, and H.L, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. of ACM SIGSAC symposium on Information, computer and communications security*, 2013, pp. 71–82.

[9] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," in *Proc. of the 31st ICDCSW*, 2011, pp. 273–281.

[10] Ayad Ibrahim, Hai Jin, Ali A. Yassin, and Deqing Zou, "Secure Rank-ordered Search of Multi-keyword Trapdoor over Encrypted Cloud Data," in *Proc. of APSCC*, 2012 IEEE Asia-Pacific, pp. 263–270.

[11] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. of IEEE INFOCOM'10 Mini-Conference*, San Diego, CA, USA, March 2010, pp. 1–5.

[12] C. Wang, K. Ren, S. Yu, K. Mahendra, and R. Urs, "Achieving Usable and Privacy-Assured Similarity Search over Outsourced Cloud Data," in *Proc. of IEEE INFOCOM*, 2012.

[13] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *RLCPS, January 2010, LNCS. Springer, Heidelberg*.

[14] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, vol.38, issue 11, pp. 39–41, 1995.

[15] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, 2003, http://eprint.iacr.org/2003/216.

[16] P. Scheuermann and M. Ouksel, "Multidimensional B-trees for associative searching in database systems," *Information systems*, vol. 7, issue 2, pp. 123–137, 1982.

[17] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. of TCC*, 2007, pp. 535–554.

[18] P. Golle, J. Staddon, and B. R. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. of ACNS*, 2004, pp. 31–45.

[19] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. of EUROCRYPT*, 2008.

[20] T. Okamoto and K. Takashima, "Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption," in *Proc. of EUROCRYPT*, 2012, pp. 591–608.

[21] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proc. of the 6th TCC*, 2009.

[22] V. Y. Lum and K. Meyer-Wegener, "An architecture for a multimedia database management system supporting content search," in *Proc. of International Conference on Computing and Information*, 1990, pp.304–313.

[23] N. Guarino, C. Masolo, and G. Verete, "OntoSeek: Content-Based Access to the Web," *IEEE Intelligent Systems*, vol. 14, no. 3, pp. 70–80, 1999.

[24] G. Varelas, E. Voutsakis, and P. Raftopoulou, "Semantic Similarity Methods in WordNet and their Application to Information Retrieval on the Web," in *Proc. of 7th ACM international workshop on Web information and data management*, 2005, pp. 10–16.

[25] P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab, "Learning taxonomic relations from heterogeneous sources," in *Proc. of the ECAI 2004 Ontology Learning and Population Workshop*, 2004.

[26] W. Wang, W. Meng, and C. Yu, "Concept Hierarchy Based Text Database Categorization in a Metasearch Engine Environment," in *Proc. of the First International Conference on Web Information Systems Engineering*, 2000.

[27] N. Nanas, V. Uren, and A. D. Roeck, "Building and applying a concept hierarchy representation of a user profile," in *Proc. of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, 2003.

[28] A. F. Gelbukh, G. Sidorov, and A. Guzman-Arenas, "Document Indexing With a Concept Hierarchy. New Developments in Digital Libraries," in *Proc. of the 1st International Workshop on New Developments in Digital Libraries*, 2001.

[29] B.B. Wang, R.I. McKay, H.A. Abbass, and M. Barlow, "Learning text classifier using the domain concept hierarchy," in *Proc. of IEEE International Conference on Communications*, 2002.

[30] Y. Chen, G.-R. Xue, and Y. Yu, "Advertising keyword suggestion based on concept hierarchy," in *Proc. of WSDM'08*, 2008.

[31] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and Application of a Metric on Semantic Nets," *IEEE Trans. System, Man, and Cybernetics*, vol. 19, pp. 17–30, 1989.

[32] A. Ralescu and A. Fadlalla, "The issue of semantic distance in knowledge representation with conceptual graphs," in *Proc. of Fifth Annual Workshop on Conceptual Structures*, 1990.

[33] D. Chen, Y. Jianzhuo, F. Liying, and S. Bin, "Measure Semantic Distance in WordNet Based on Directed Graph Search," in *Proc. of EEEE'09*, 2009, pp. 57–60.

[34] ODP, http://www.dmoz.org/Science/.

[35] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. of SIGMOD*, 2009, pp. 139–152.

[36] W. W. Cohen, "Enron email dataset," http://www.cs.cmu.edu/∼enron/.

[37] J. Cao, P. Karras, P. Kalnis, and K.-L. Tan, "Sabre: a sensitive attribute bucketization and redistribution framework for t-closeness," *VLDB J.*, vol. 20, no. 1, pp. 59–81, 2011.

[38] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement," *IEEE Trans. on Parallel and Dist. Systems*, DOI: 10.1109/TPDS.2015.2506573.

[39] B. Yao, F. Li, X. Xiao, "Secure nearest neighbor revisited," in *Proc. of ICDE 2013*, PP. 733–744.

[40] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, Ma. Rosu, M. Steiner, "Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries," in *Proc. of CRYPTO 2013*, pp. 353–373.

[41] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu,"Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing," *IEICE Transactions on Communications*, 2015, vol. E98-B, no. 1, pp.190-200.

[42] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, and S. Bellovin, "Blind Seer: A Searchable Private DBMS," in *Proc. of IEEE S&P 2014*, pp. 359–374.

[43] C. Chen, X. Zhu, P. S, J. Hu, S. Guo, Z. tari, and A. Y. Zomaya, "An efficient privacy-Preserving Ranked Keyword Search Method," *Parallel and Distributed Systems,IEEE Transactions on*, 2015.

[44] Z. Fu, F. Huang, X. Sun, A. V. Vasilakos, and C. Yang, "Enabling Semantic Search based on Conceptual Graphs over Encrypted Outsourced Data," *Services Computing,IEEE Transactions on*, 2016. DOI:10.1109/TSC.2016.2622697.

[45] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," *Proceedings of the 2012 ACM conference on Computer and communications security*, pp.965-976,2012.

[46] M. Chase, and S. Kamara, "Structured encryption and controlled disclosure," *In Advances in Cryptology-ASIACRYPT 2010,Springer Berlin Heidelberg*, pp.577-594,2010.

[47] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Towards Efficient Multi-keyword Fuzzy Search over Encrypted Outsourced Data with Accuracy Improvement", *IEEE Transactions on Information Forensics and Security*, vol.11,no.12, pp.2706-2716,2016. DOI:10.1109/TIFS.2016.2596138.

[48] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A Privacy-preserving and Copy-deterrence Content-based Image Retrieval Scheme in Cloud Computing," *IEEE Transactions on Information Forensics and Security*, 2016. DOI:10.1109/TIFS.2016.2590944.

[49] Y. Ren, J. Shen, J. Wang, J. Han, and S. Lee, "Mutual Verifiable Provable Data Auditing in Public Cloud Storage," *Journal of Internet Technology*, vol.16,no.2, pp.317-323,2015.

[50] B. Gu, and V. S. Sheng, "A Robust Regularization Path Algorithm for -Support Vector Classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2016. DOI:10.1109/TNNLS.2016.2527796.

[51] J. Wang, X. Chen, and X. Huang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Transactions on Computers*, vol.64,no.11, pp.3293-3303,2015.

[52] F. Cheng, Q. Wang, and Q. Zhang, "Highly Efficient Indexing for Privacy-Preserving Multi-keyword Query over Encrypted Cloud Data," *International Conference on Web-Age Information Management*, pp.348-359,2014.

[53] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang. "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Transactions on Information Forensics and Security*, vol.12,no.8, pp.1874-1884,2017.

[54] X. Chen, J. Li, and J. Ma, "New algorithms for secure outsourcing of modular exponentiations," *Parallel and Distributed Systems,IEEE Transactions on*, vol.25,no.9, pp.2386-2396,2014.

[55] Z. Fu, X. Wu, Q. Wang, and K. Ren. "Enabling central keyword-based semantic extension search over encrypted outsourced data," *IEEE Transactions on Information Forensics and Security*, vol.12,no.12, pp.2986-2997,2017.

[56] J. Li, J. Li, and X. Chen, "Identity-based encryption with outsourced revocation in cloud computing," *Computers,IEEE Transactions on*, vol.64,no.2,pp.425-437,2015.

[57] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang. "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol.11,no.4, pp.789-798,2017.

[58] P. Li, J. Li, Z. Huang, C. Gao, W. Chen, and K. Chen. "Privacy-preserving outsourced classification in cloud computing", *Cluster Computing*,2017:1-10. DOI: 10.1007/s10586-017-0849-9.

[59] J. Li, X. Chen, F. Xhafa, and L. Barolli. "Secure deduplication storage systems supporting keyword search". *Journal of Computer and System Sciences*, vol.81,no.8, pp.1532-1541, 2015.

[60] Kiayias, A., Oksuz, O., Russell, A., Tang, Q., and Wang, B. "Efficient Encrypted Keyword Search for Multi-user Data Sharing," *Proc. of European Symposium on Research in Computer Security 2016*, pp.173-195,2016. Springer International Publishing.

[61] Z. Fu, X. Sun, S. Ji, and G. Xie, "Towards efficient content-aware search over encrypted outsourced data in cloud," *Proc. of IEEE INFOCOM 2016*, pp.1-9,2016.

[62] J. Li, Y. Zhang, X. Chen, and Y. Xiang. "Secure attribute-based data sharing for resource-limited users in cloud computing," *Computers and Security*, vol.72, pp.1-12, 2018. DOI:10.1016/j.cose.2017.08.007.

Zhangjie Fu received his PhD in computer science from the College of Computer, Hunan University, China, in 2012. He is currently an Associate Professor at School of Computer and Software, Nanjing University of Information Science and Technology, China. He was a visiting scholar of Computer Science and Engineering at State University of New York at Buffalo from March, 2015 to March, 2016. His research interests include Cloud Security, Outsourcing Security, Digital Forensics, Network and Information Security. His research has been supported by NSFC, PAPD, and GYHY. Zhangjie is a member of IEEE and a member of ACM.

Lili Xia received her BE in Soft Engineering from Nanjing University of Information Science and Technology (NUIST), Nanjing, China, in 2016. She is currently pursuing her MS in computer science and technology at the Department of Computer and Software, Nanjing University of Information Science and Technology, China. Her research interests include cloud security and information security.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2819121, IEEE Transactions on Information Forensics and Security

Xingming Sun received his BS in mathematics from Hunan Normal University, China, in 1984; his MS in computing science from Dalian University of Science and Technology, China, in 1988; and his PhD in computing science from Fudan University, China, in 2001. He is currently a professor at the Department of Computer and Software, Nanjing University of Information Science and Technology, China. In 2006, he visited the University College London, UK; he was a visiting professor in University of Warwick, UK, between 2008 and 2010. His research interests include network and information security, database security, and natural language processing.

Alex X. Liu received his Ph.D. degree in Computer Science from The University of Texas at Austin in 2006. He received the IEEE & IFIP William C. Carter Award in 2004, a National Science Foundation CAREER award in 2009, and the Michigan State University Withrow Distinguished Scholar Award in 2011. He is an Associate Editor of IEEE/ACM Transactions on Networking, an Associate Editor of IEEE Transactions on Dependable and Secure Computing, and an Area Editor of Computer Communications. He received Best Paper Awards from ICNP-2012, SRDS-2012, and LISA-2010. His research interests focus on networking and security.

Guowu Xie received his Bachelor degree from Tongji University, China, in July 2005; received his MS in electrical engineering from Shanghai Jiao Tong University, China, in 2008; obtained his PhD in computer science from the department of computer science and engineering, University of California, Riverside,USA, in 2012. Currently, he works as a Research Scientist in Facebook, USA. His research interests include network traffic monitoring and analyzing, information security, cloud security.