Copyright © 2019 Tech Science Press CMC, vol.59, no.2, pp.473-491, 2019

Privacy-Preserving Content-Aware Search Based on Two-Level Index

Zhangjie Fu[1, *], Lili Xia[1], Yuling Liu[2] and Zuwei Tian[3]

Abstract: Nowadays, cloud computing is used more and more widely, more and more people prefer to using cloud server to store data. So, how to encrypt the data efficiently is an important problem. The search efficiency of existed search schemes decreases as the index increases. For solving this problem, we build the two-level index. Simultaneously, for improving the semantic information, the central word expansion is combined. The purpose of privacy-preserving content-aware search by using the two-level index (CKESS) is that the first matching is performed by using the extended central words, then calculate the similarity between the trapdoor and the secondary index, finally return the results in turn. Through experiments and analysis, it is proved that our proposed schemes can resist multiple threat models and the schemes are secure and efficient.

Keywords: Semantic search, two-level index, expanded central-keyword.

1 Introduction

Today, heterogeneous Internet of Things is applied popularly [Qiu, Chen and Li (2018)]. Zhang et al. [Zhang, Li and Dai (2018)] proposed a public verifiable outsourcing scheme based on matrix multiplication in the Internet of Things environment. Many people are preferring to storing data in the cloud. Yang et al. [Yang, Xu and Weng (2018)] provides a lightweight proof of storage for privacy protection. To utilize the data in the cloud efficient, cloud computing is widely developed. Cloud computing, not only reducing the local data maintenance costs, but also provide simple and efficient calculations. In addition, cloud computing gives a convenient way to share resources between data owners and legal data users.

Some scheme [Li, Liu and Chen (2015)] can protect the security of data search at some degree. However, cloud servers are not completely honest. Cloud servers enforce system protocols and capabilities honestly, but they guess the content of important files stored on them actively. The existing solution is to encrypt the data before uploading. However, how to effectively search encrypted data is an important issue.

At present, there are too many researches on ciphertext search by domestic and foreign

[1] School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing, 210044, China.

[2] College of Computer Science and Electronic Engineering, Hunan University, China

[3] School of Information Science and Engineering, Hunan First Normal University, Hunan, China.

[*] Corresponding Author: Zhangjie Fu. Email: wwwfzj@126.com.

experts. Song et al. [Song, Wagner and Perrig (2002)] proposed a searchable encryption method which must encrypt the data set for each query, in each query, the data set needs to be fully scanned make the computational consumption enormous. At the same time, this method cannot resist statistical analysis attacks. Dan et al. [Dan, Crescenzo and Ostrovsky (2004)] proposed a keyword search scheme based on public key encryption. This scheme is also aimed at the matching of keywords and ciphertexts, but the scope of utilization is too small. After that, Li et al. [Li, Wang and Wang (2014)] proposed a wildcard-based ciphertext search method to achieve fuzzy search to a certain degree, but it needs the support of the semantic library. A new scalable dynamic access scheme is proposed in Weng et al. [Weng, Weng, and Zhang (2018)]. At present, most of the research on index-based ciphertext search methods use tree index structure to improve the search efficiency. Fu et al. [Fu, Sun and Liu (2015)] proposed a search scheme to improve efficiency by using k-d tree. Leslie et al. [Leslie, Jain and Birdsall (1995)] proposed an efficient query scheme based on multi-dimensional B tree. Ciaccia et al. [Ciaccia, Patella and Rabitti (1997)] used M-tree to construct index metric space which improved search efficiency. Kurasawa et al. [Kurasawa, Takasu and Adachi (2008)] proposed a Peer-to-Peer (P2P) information search scheme based on Huffman distributed hash table. The index structure was changed by Huffman coding to reach the load balancing. However, the data preprocessing consumes too many computing and storage resources. Xu et al. [Xu, Li and Dai, (2019)] proposed an efficient geometric range query scheme.

In this paper, we put forward the privacy-preserving content-aware search by using the two-level index (CKESS). We build a novel two-level index framework based on the existed forward and reverse indexes, which can improve the search efficiency. Simultaneously, for improving the search precision, our program has expanded the central word of the trapdoor. The basic framework of privacy-preserving content-aware search by using the two-level index has been proposed firstly. And we propose two search schemes that can resist different threats. Followings are the main contributions of the paper.

1. We propose the privacy-preserving content-aware search by using the two-level index (CKESS) to reduce the impact of the number of files on the search time. The difference from the existed index is matching the trapdoor with the first-layer index instead of the file set, which greatly improves our search efficiency;
2. In order to be close to the user's semantic requirements, we have semantically extended the central keyword of the trapdoor based on the two-level index, thus improving the search precision. Our scheme first uses the extension of the central word of the trapdoor to match the first index. After that, the similarity between the secondary index and the trapdoor keyword is calculated, and finally the result is returned in order.
3. Through experiments and analysis, it shows that our scheme can effectively resist different threat attacks and prove the efficiency, accuracy and security of the scheme.

The following are the arrangements for the other sections of this paper. In the Section 2, we introduce the related work. We arrange the system model, threat model, and implementation goals of the program in the third section. The fourth part is mainly to describe the main content of the index construction and schemes implementation. The fifth part analyzes the security of the program. The final part is the experimental analysis and the summary of the article.

2 Relation work

2.1 Multi-keyword search

In 2011, Cao et al. [Cao, Wang and Li (2014)] first proposed a secure multi-keyword search solution (MRSE) that resisted two threat models (known ciphertexts and known backgrounds). The solution used vector space model and secure inner product to sort after the search results. In 2014, they proposed improvements and added TF / IDF algorithm to improve the search accuracy. Yang et al. [Chen, Huang and Li (2015)] proposed to use the trapdoor concept to search all encrypted documents. Fu et al. [Fu, Sun and Linge (2014)] proposed multi-keyword sorting search scheme supporting synonym query in cloud environment. These programs to achieve a multi-keyword search, expanded ciphertext search capabilities. In terms of efficiency and accuracy of multi-keyword search, Chen et al. [Liang, Huang and Guo (2016)] used sparse matrices to achieve a safe large-scale linear equation, further improving the retrieval efficiency. In 2016, Liang et al. [Li, Yang and Luan (2016)] proposed that search based on regular language is more efficient than other search schemes. In the same year, Li et al. [Wang, Li and Wang (2015)] proposed a multi-keyword search based on the fine-grained, which improved the accuracy and efficiency of the search scheme through sub-dictionaries and preference factors. To combat the Selective Chosen-Plaintext Attacks, Wang et al. [Li, Liu and Wang (2016)] proposed a SE-based range search in 2015. In 2016, Li et al. [Chuah and Hu (2011)] proposed the first range query solution that can defend against selective keyword attacks (IND-CKA). Li et al. [Li, Chen and Chow (2018)] proposed a privacy-aware multi-keyword attribute encryption scheme that hides attribute information in ciphertext and allows tracking of non-honest users who have keys. Combining kNN and attribute encryption, a dynamic search symmetric encryption scheme is proposed in Li et al. [Li, Yang and Dai (2017)]. Gao et al. [Gao, Cheng and He (2018)] proposed to combine the double-blind technology with the addition of homomorphic encryption to realize the privacy protection of both parties. Li et al. [Li, Liu and Dai (2018)] proposed an efficient searchable symmetric encryption scheme in mobile cloud environment.

2.2 Semantic search

Li et al. [Li, Wang and Wang (2014)] propose a single fuzzy keyword search scheme by using edit distance. Chuah et al. [Chuah and Hu (2011)] implement fuzzy search based on pre-defined word set as a whole, and propose a fuzzy search scheme to achieve secure search for multi-keywords. Kuzu et al. [Kuzu, Islam and Kantarcioglu (2012)] designed a similar search scheme using minhash based on Jaccard distance. Fu et al. [Fu, Shu and Wang (2015)] proposed similarity search scheme for encrypted documents based on simhash. Wang et al. [Wang, Yu and Zhao (2015)] proposed a fuzzy search based on multi-keywords for achieving range query using a sequence reservation encryption and two-layer Bloom. Huang et al. [Huang, Fu and Sun (2016)] combined gene sequences with secure searchable indexes to propose a secure searchable encryption scheme. Our earlier work [Fu, Wu and Guan (2016)] used a new keyword transformation to get a better accuracy. Fu et al. [Fu, Wu and Wang (2017)] improved accuracy by using center word expansion. Fu et al. [Fu, Xia and Sun (2018)] proposed the conceptual hierarchical semantic search scheme under dual servers.

3 Problem formulation

3.1 System model

Fig. 1 shows that our system model is mainly composed of the following three entities.

Data owner: The data owner is the person who owns the source data. To facilitate search, they need to build efficient indexes based on the data. Finally, the source data and the index are encrypted and uploaded to the cloud server.

Cloud server: Storing and managing the encrypted data received are the main work of the cloud server. Upon receiving the user's retrieval request, the data is processed and the results are returned in turn.

Data user: The data user who received a secure authorization certificate processes the query to generate a corresponding trapdoor and uploads it to the cloud.

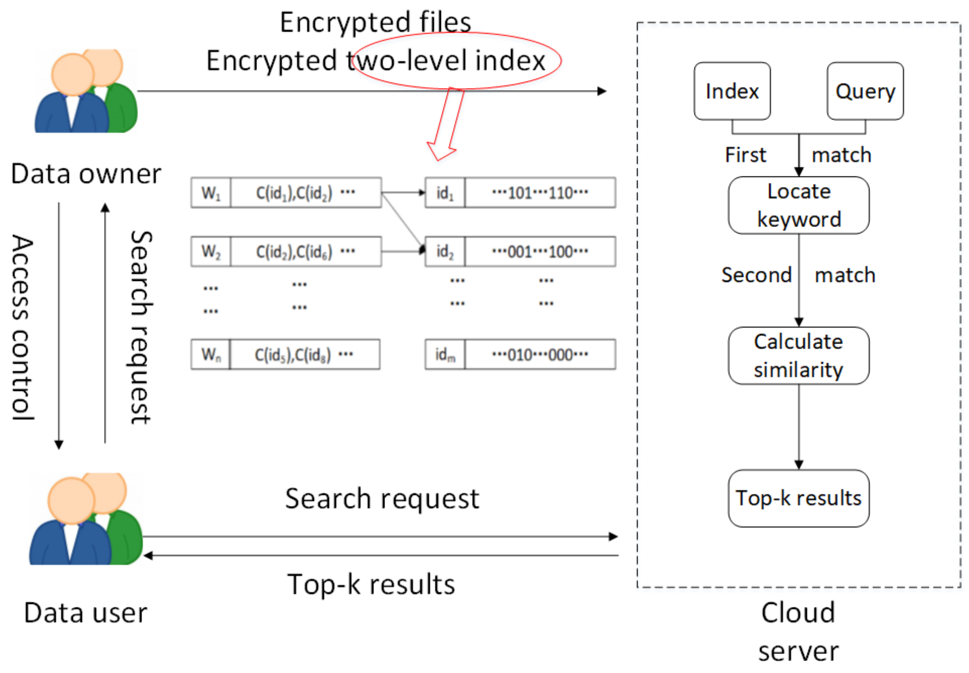

Figure 1: The system model

3.2 Threat model

In the overall system model, we believe the data owner and user are honest and trustworthy, but the cloud server is not completely trusted. As the references describe [Cao, Wang and Li (2014)], the cloud server can execute programs accurately without tampering with the DO and DU instruction sets; however, it can collect keyword information and deduces the contents of the file by analyzing the relationship between trapdoor functions. Therefore, according to different privacy requirements, our threat model is as follows.

Known Ciphertext Model: The uploaded information of files, indexes, and trapdoor is easy to get for the cloud server, but it does not understand the meaning.

Known Background Model: Apart from file data, indexes, and trapdoors, the cloud also understands some background information, including file statistics, keyword frequency, and the relationship between keywords and query. With this extra information, the cloud server can speculate on some ciphertext information.

3.3 Design goal

Due to the above threat models, the design goals of this paper are as follows:

1. Scalability. We hope to implement the expansion of the central keywords, that is to say, after the user enters the search keywords, the center word can be found quickly and located accurately, and the semantic expansion of the central keyword can be achieved. The result returned must be related to the central word and its extension.
2. Efficiency. Considering the data processing of cloud servers, our approaches are designed that document size has independence. In simple terms, the relationship between the search time and fileset size should be sub-linear.
3. Privacy protection. Our schemes should provide privacy protection of data information of filesets or keywords during cloud search.
4. Results accuracy. This paper focuses on semantic search, so the accuracy of the search results is extremely important. Our solution should improve the retrieval accuracy as much as possible to achieve a higher standard.

3.4 Preliminaries

Stanford Parser. Stanford parser is basically a lexicalized probability context-free parser, and also uses dependency analysis. Different analysis results can be output according to different grammatical points. The parser uses the knowledge of the language obtained from the parsing of the sign language to get the appropriate analysis of the new sentence.

WordNet. WordNet is a huge English vocabulary database [Miller (1990)]. Nouns, verbs, adjectives and adverbs are each organized into a network of synonyms. Each synonym set represents a basic semantic concept, and these sets are also connected by various relationships. The resulting meaningful vocabulary and conceptual network can be browsed through the browser.

Locality-Sensitive Hashing. The local sensitive hash *(LSH)* algorithm [Har-Peled, Indyk and Motwani (1998)] is mainly used to solve the nearest neighbor search problem efficiently. The basic idea of this algorithm is to map datasets to different collision buckets of multiple hash tables through a set of hash functions that meet certain constraints, then establishing multiple hash tables. So, under certain similarity measures, the points which are closer to the same conflict bucket (the higher similarity) have the greater probability, and the points which are farther to the same conflict bucket (the lower similarity) have the smaller probability. In this way, the data set is divided into multiple sub-sets. The data in each sub-set is adjacent and the number of the sub-set is small. So the problem of finding adjacent elements in a very large set is transformed to find adjacent elements within a small set. This method reduces the amount of data to be compared greatly when we do a search. We deem that a hash function family H is (d_1, d_2, p_1, p_2)-sensitive if for random inputs of x, y, and $h \in H$ satisfy the following:

$$\text{If } d(x, y) \leq d_1 : \Pr\left[h(x) = h(y)\right] \geq p_1 \tag{1}$$

$$\text{If } d(x, y) \geq d_2 : \Pr\left[h(x) = h(y)\right] \leq p_2 \tag{2}$$

Where $d(x, y)$ denotes the distance between x and y, $h(x)$ and $h(y)$ denote the hash function of x and y, d_1 and d_2 mean the distance thresholds, p_1 and p_2 mean the probability threshold, and $d_1 < d_2, p_1 < p_2$.

4 Secure search scheme

4.1 Main innovative idea

In order to achieve a secure and efficient encryption search scheme, three important design choices must be made, all of which are closely related and to some extent determine the performance of the search solution. First, the data structure used to construct the file index; second, an efficient search scheme that enables a good match between the query keywords and the file index; and finally, a secure privacy mechanism that combines with the first two design choices to protect the privacy of files and retrieving privacy.

Based on the above design choices, we propose privacy-preserving content-aware search based on two-level index. A secure two-level index for the file set is built and encrypted before uploading to the cloud. The trapdoor is constructed on the data user side and the data user provides the query keyword. We first locate the central word of the query keyword, then extend the central keyword, and upload the extended keywords to the cloud encryption. The mainly work of the cloud server is searching. When the cloud server gets the trapdoor from the user, in the first-level matching, the cloud server matches the extended keyword with the index keyword. Without loss the search accuracy, the search range is reduced. In the second-level matching, the other keywords in each file are matched with other non-central keywords in the trapdoor. Finally, the results are sorted and returned to the data user.

In this section, we detail the scenario. We first introduce the central keyword expansion algorithm in plaintext mode and the two-level index construction algorithm, and then propose the complete schemes in ciphertext mode.

4.2 Central keyword extension

In the actual situation, the user usually inputs a string of keywords to make a query, and when the user searches different targets, the weight of each keyword is different. There have been many achievements in the study of the importance of keywords. [Li, Yang and

Luan (2016)] use a method of super-increasing sequence to order the keywords. In this way, getting the order of the query keywords' importance is easily. However, this method requires the user to sort the keywords according to the user's expected value of the keyword. The disadvantage of this approach is that defaulting importance to the order of all keywords. Our early work Fu et al. [Fu, Ren and Shu (2016)] proposed to establish a unique interest model for different users, and use the grammatical relationship tree to measure the weight of keywords input by the user. In this way, a search that satisfies the needs of the user is realized.

Definition of relationship: The relationship contained in the keyword reflects the user preference information. Keywords with more semantic relationships necessarily reflect more user information, so the weight of the keyword is heavier. So, we have the definition 1.

Definition 1: For each keyword, we assign it an initial importance of 1. Assuming that the keyword has grammatical relationships with another keyword, we change its value to 1+R (relation), where R represents the total importance of the relationship between two keywords.

The semantic relationship of keywords is expressed as a connection between different words on the grammatical relationship tree. The shorter the distance of this connection, the more similar the meaning of the words. Therefore, we can quantify the importance of the grammatical relationships between keywords by using the distance between the keywords in the grammatical tree. However, the relationship is mutual, so the value of the importance should be divided into two parts. So we have the definition 2.

Definition 2: For the two keywords w_1 and w_2, there is a semantic relation A between them, and the interest preferences of the user included in A is $R(A)=1/\ln(d)$, where d represents the distance of the keywords in the semantic relation tree. The increased importance of w_1 is $d_1\cdot\ln(d)/d$, and $d_2\cdot\ln(d)/d$ is the increased importance of w_2, where d_1 and d_2 are the distance between the keywords and their ancestor, and the total between d_1 and d_2 is d.

Note that, although prepositions are often used in articles and phrases to indicate the relation between two keywords, prepositions cannot be an independent part of a sentence. Therefore, when we calculate the importance of grammatical relations, such keywords do not be included.

Definition 3: For an independent query Q, the total weight of the grammatical relation is n, where n represents the number of query keywords contained in this query. For a keyword w, $p\times n$ is its weight, in which p means the percent of the importance of the keyword w in the query Q. So we can get the keyword weight of the keyword w as follow

$$KW(w)=p\times n=\left(1+\sum R\right)\times n/\sum_{i=1}^{n}\left(1+\sum R\right) \tag{3}$$

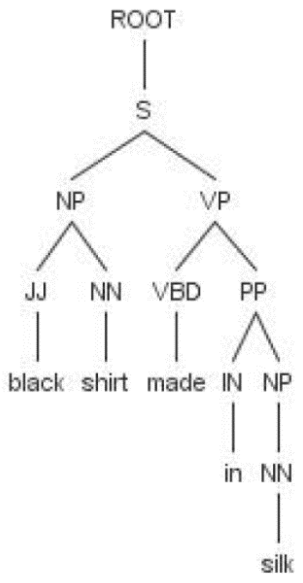

Figure 2: The grammatical relationship tree of "black shirt made in silk"

Decision of central keyword: To represent the grammatical relations, we use the Stanford parser in this paper. The function of the Stanford parser is to transform keywords into a grammatical relationship tree. Through this tree, the grammatical relationship between words is clearly expressed. For instance, Fig. 2 shows the grammatical relationship tree of "black shirt made in silk", in which "NP" represents the noun phrase, "NN" represents the noun, and "JJ" represents an adjective or a number, and so on.

We calculate the distance d between two keywords in the grammatical tree to indicate the importance of the relation between two keywords. As the Fig. 2 shown, the distance between "black" and "shirt" is 4, so $R(a\text{mod}) = 1/\ln(4)$. Since each relation is shared by two keywords, we divide each relationship into two parts according to the following rules: $R_1 = (d - d_1) * R / d$, $R_2 = 1 - R_1 = (d - d_2) * R / d$, where d_1 is the distance between the first keyword and the ancestors of the two keywords, as the same, the distance between the second keyword and the ancestor of the two keywords is d_2. Since the weight of each keyword has been calculated by this way, we choose the keywords which have the highest weight as the central keywords.

Extension of central keywords: For the same meaning, there may be many different words to express, for example, "blouse" and "shirt" mean the same thing. When the user queries one of the words, if the word is not in the server, the result will not be returned. In this case, the user needs to change a synonym and search again, which wastes the search cost. Existing solutions either increase the index build cost or extend the keyword in a small range.

Therefore, we calculate the weight of keyword to get the central keyword. Then, we expand the central keyword. In this way, we not only ensuring the results of user search, but also ensure that trapdoor generated efficiency. So, we should balance the relationship between efficiency and functionality.

Definition 4: For a query Q, we determine that the keyword W (W belongs to Q) as the central keyword when the weight of the keyword is greater than the weight of others.

We extend the keywords by using WordNet. WordNet contains a huge amount of English words [Lin (1998)]. In this vocabulary, words with the same meaning are grouped and called synonyms. So, when we search for "pants", not only will we return "pants", but we will also return files related to the "trousers".

We use the central word as the benchmark to calculate the weight of each extended word. The higher the weight, the closer the meaning of the extended word, and the higher the quality of the expansion.

We use Lin's measure [Wong, Cheung and Kao (2009)] to calculate the similarity between the keywords A and B, and the calculated similarity is:

$$Sim(w_1, w_2) = [2 \times I(F(W_1) \cap F(W_2))] / [I(F(W_1) + F(W_2))] \tag{4}$$

Where $F(w)$ represents the feature set of W. $I(S) = -\sum_{f \in S} \log P(f)$ represents the information contained in a set of features S, where $P(f)$ is the probability of feature f. When two words have the same feature set, the similarity range from 0 to 1, in which 0 means two words have no same feature, and 1 means two words have the same feature set.

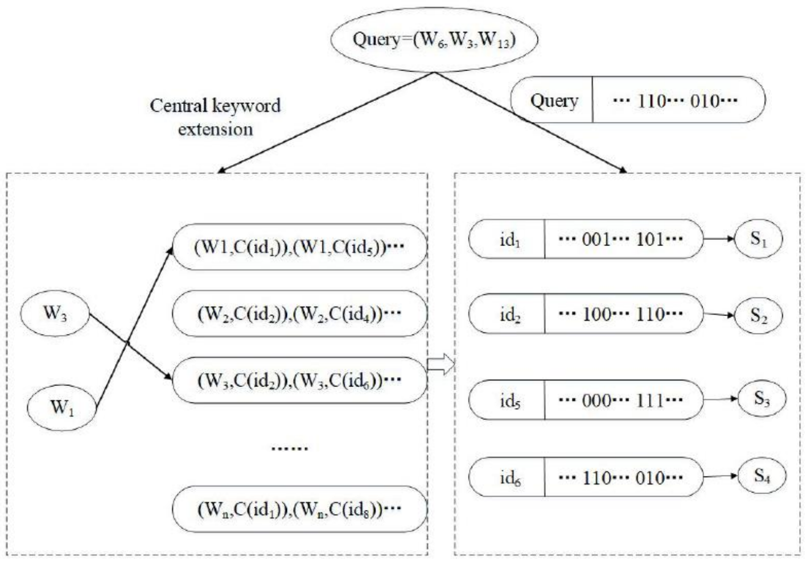

Figure 3: The framework of our scheme

4.3 Two-level index framework

Most of the existing index construction methods build reverse indexes to facilitate searching. However, this index construction method still has a lot of space for improvement in efficiency and precision. Therefore, this paper proposes a new search index called two-level index. This index construction method is mainly to index the file set twice with different tags. The construction of the first-layer index is based on keywords, and the index of keyword to file set is constructed; the second-layer index is based on a single file, and the index of a single file to keyword set is constructed.

For improving efficiency and accuracy, our index is divided into two steps when matching. First, the keyword expanded by the central word is matched with the first layer index, thereby narrowing the file range, and then the similarity calculation is performed on the filtered file and the trapdoor, and the result is sequentially returned to the user.

As shown in Fig. 3, by matching the extended keyword (w_1, w_3) with the first-layer index, the file set that needs to be further matched is narrowed down to $(i.e.C(id_1), C(id_2), C(id_5), C(id_6))$. After that, the similarity calculation is performed on each file and the trapdoor in the second-layer index. As shown in Fig. 3, S_1, S_2, S_3 and S_4 are the final similarity scores, which is sorted and returned to the user. The higher the score, the more consistent with the user's needs.

4.4 CKESS₁ scheme: secure scheme under the known ciphertext model

Our solution combines two-level index and central keyword expansion on the basis of MRSE [Cao, Wang and Li (2014)]. We describe our scheme under the known ciphertext model in detail in this section. We extend each vector to $(m+2)$ dimensions. The extended dimension adds complexity to the index, making it difficult for cloud servers to guess trapdoors and file information. The main algorithms of our scheme are shown as follow.

KeyGen: Given the security parameter λ and the file set F with the keyword set W, the key SK is generated safety. The data owner generates the SK(S, K_1, K_2, M_1, M_2). We get two random sequences $K_1, K_2 \leftarrow \{0,1\}^{\lambda}$. (M_1, M_2) are two $(m+2)\times(m+2)$ invertible matrices and $S \in \{0,1\}^m$ is a vector.

Index Construction: For a file set F given a set of keywords W, the data owner initializes I_1 and I_2 to an empty array in order to obtain the inverted index I_1 and the forward index I_2 at the same time. Then, for each of the keywords in W, the data owner uses the key to encrypt the file identifier, and the LSH is used in the Bloom filter. The data owner encrypts the file ID and appends them to set I_1 and then sets all positions in I_2 to 1. Finally, encrypting the Bloom filter in the secondary index I_2 using secure kNN encryption with (M_1, M_2, S) [Yang, Liu and Li (2015)]. The index I_2 is divided into two vectors $I_2^{'}, I_2^{''}$: For each element $i_j \in I$, let $i_j^{'} = i_j^{''} = i_j$,if $s_j \in S$ equals 1;

otherwise $i_j^{'} = \frac{1}{2} i_j + r, i_j^{"} = \frac{1}{2} i_j - r$ where r is a random number, then I_2 is represented as $\left\{ M_1^T \cdot I_2^{'}, M_2^T \cdot I_2^{"} \right\}$. At last, the data owner gets Index = I_1, I_2 and uploads the index and the encrypted file set to the cloud.

Trapdoor Generation: The user inputs the query Q, and performs a weight calculation for Q to obtain the central word. Then uses WordNet to expand the central word. So, the user gets the extension query EQ. The data user encrypts EQ and builds a Bloom filter Bf_q for $\vec{w}$. After that, we divide the Bloom filter Bf_q into two vectors $Bf_q^{'}$, $Bf_q^{"}$: If $s_j \in S$ equals 0, then set $Bf_q^{'} = Bf_q^{"} = Bf_q$; otherwise, $Bf_q^{'} = \frac{1}{2} Bf_q + r', Bf_q^{"} = \frac{1}{2} Bf_q - r'$, Where r' is another random number. Finally, token $= EQ, \left\{ M_1^{-1} \cdot Bf_q^{'}, M_2^{-1} \cdot Bf_q^{"} \right\}$, K is obtained and uploading to the cloud, where K is the result that K_2 does the pseudo-random function.

Search: The cloud server uses the index to match the trapdoor. First, the cloud server get the encryption identifiers in I_1, and matches the EQ with the I_1 to get the intermediate results. Then, Token is decrypted by K, and retrieved the Bloom filter I_2 = $\left\{ M_1^T \cdot I_2^{'}, M_2^T \cdot I_2^{"} \right\}$. And the cloud server obtains the secret key K but it does not disclosure any information except the search result of EQ. This leakage can be acceptable because the search result of EQ are not the sensitive data. It is easy to get the inner product between Bf_q and I_2.

$$Bf_q \bullet I_2 = M_1^T I_2^{'} \cdot M_1^{-1} Bf_q^{'} + M_2^T I_2^{"} \cdot M_2^{-1} Bf_q^{"} = I_2^T \cdot Bf_q \tag{5}$$

Then, the server returns the results based on the inner product to the user in turn.

4.5 CKESS$_2$ scheme: Secure scheme under the known background model

Under this model, the cloud server not only knows the information, but also has a certain understanding of the background environment. Through background information such as text analysis and word frequency, the cloud server can speculate on trapdoors and indexes. In order to resist this threat, we propose the following scheme.

KeyGen: The algorithm is basically the same as the previous one. The only difference is that k hash keys $(key_1, key_2, \cdots key_k)$ are inserted into SK. We get SK= $(S, K_1, K_2, key_1, key_2, \cdots key_k, M_1, M_2)$.

Index Construction: Due to the change of the SK, the secret key need to re-generated and the vector S_w of each keyword should be calculated again. Then, data owner encrypts the Bloom filter in the secondary index I_2 using secure kNN encryption with (M_1, M_2, S_w). The index I_2 is divided into two vectors $I_2^{'}, I_2^{''}$ according to the same rules. So, I_2 is represented as $\{M_1^T \cdot I_2^{'}, M_2^T \cdot I_2^{''}\}$. At last, the data owner gets Index= I_1, I_2 and uploads the index and file set to the cloud.

Trapdoor Generation: Compared with the CKESS$_1$, the difference in the algorithm is that the pseudo-random function is modified by using k hash keys, and s_j is changed into S_w. Finally, we represent the token as $EQ, \left\{M_1^{-1} \cdot Bf_q^{'}, M_2^{-1} \cdot Bf_q^{''}\right\}, K$.

Search: In this algorithm, the cloud server compute matches the EQ with the I_1 firstly, and obtains the intermediate results. Then, according the intermediate results, the cloud server compute the similarity score between the I_2 and the Bf_q. At last, the cloud server returns the top-k results to data user.

5 Security analysis

This paper is mainly to conduct security analysis from the following aspects.

Privacy: In this paper, the documents need to be encrypted before uploading to the cloud. We make use of the secure encryption framework MRSE to encrypt the documents [Cao, Wang and Li (2014)]. So, our data privacy is protected well.

This paper uses pseudo-random functions, secure encryption algorithms and KNN algorithm to encrypt keywords, file sets and Bloom filters securely. This prevents the Plaintext Leakage. Due to the increase of random numbers, the cloud server cannot know any trapdoor information (whether different trapdoors are generated by the same search request), except that different trapdoors retrieve the same keyword at the first search. At the same time, in the second search, all query keywords are used to match, so the cloud server cannot know the query results.

Confidentiality: Since the index and trapdoors are encrypted in the cloud server, they are not obtained by the cloud server. This encryption algorithm has been proved to be safe in the known ciphertext model [Yang, Liu and Li (2014)]. However, we added k hash keys to improve the encryption algorithm for security under the known background model.

Unlinkability: We introduced k hash keys when splitting the vector, so that the same search request can get different trapdoors. And the random hash key makes the similarity scores different. Trapdoor unlinkability can be guaranteed by using this method.

6 Performance analysis

In this section, we experimented with different scenarios to verify performance. We do these experiments under Windows 7 by using Java language, and the specific CPU parameters is Core 2 CPU 2.93 GHz.

6.1 Precision

In order to improve the semantic precision of the scheme, we have expanded the keywords. In order to evaluate the precision of the scheme, two probabilities have been set, pr1 and pr2, where pr1 represents there is single-character different between the keywords, and pr2 is multi-character. As the Fig. 4 shown, when c is the same, pr1 decreases with the number of functions increases, and pr2 is reversed. When the number of functions is constant, pr1 increases as c increases, and pr2 is reversed. So we need to strike a balance between pr1 and pr2.

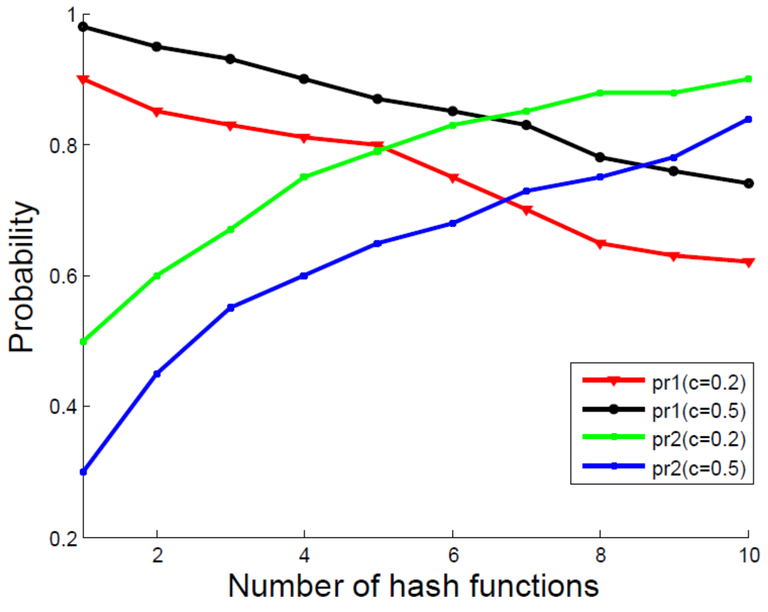

Figure 4: Precision of scheme

6.2 Index construction

When constructing and encrypting index vectors, the dimension of the index vector is the important factor that affect the index constructing. As shown in Fig. 5, because we use the high-dimensional submatrix to build the secret matrix when encrypting the index, this makes our index construction time much better than MRSE. In addition, there is a positive linear relationship between the index build time and the number of file sets. Since the dimension of $CKESS_2$ is more than $CKESS_1$, the build time is higher than $CKESS_1$.

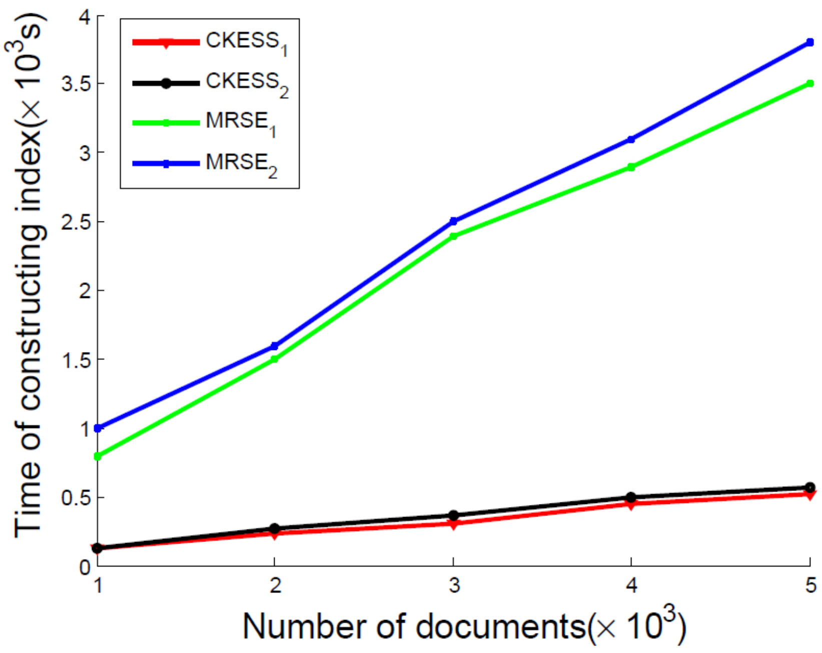

Figure 5: The time of Index construction

6.3 Query generation

The time of trapdoor generation mainly includes the confirmation of the central keyword, the extension of the central word, the calculation of similarity, the construction of the query vector and the encryption of query vector. As seen from Fig. 6, the build time of MRSE is not increase as the query keyword increases, but our schemes are opposite. The main reason is that our schemes contain the time of expanding the central keyword. With the increase of query keywords, the time taken for the expansion of the central word is also increase. And due to the high dimension, CKESS$_2$ cost more time than CKESS$_1$.

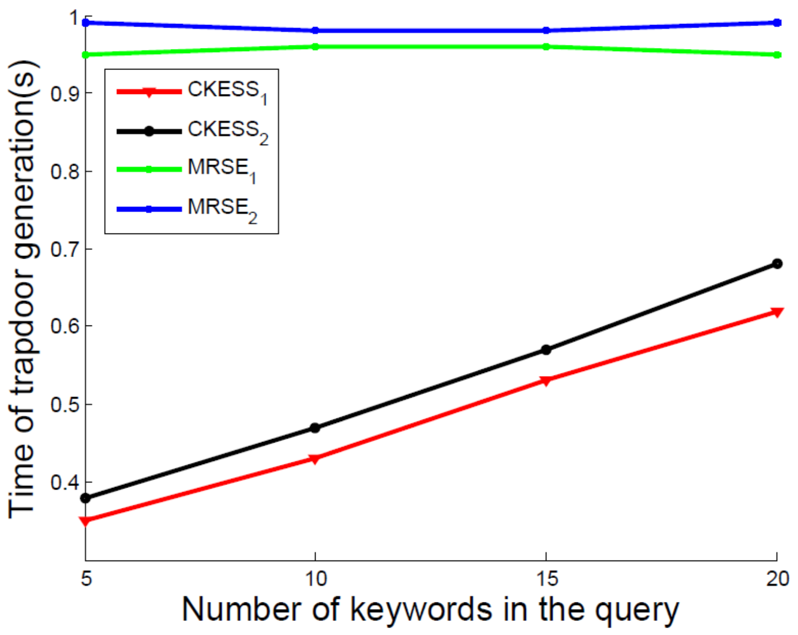

Figure 6: The time of query generation

6.4 Search efficiency

As the most important performance indicator of the schemes, Fig. 7 shows the search time of the scheme. From Fig. 7, we can see that with the increase of file sets, MRSE are linearly increase, but CKESS are increased slowly. The reason for it is that MRSE traverse all files in the cloud for each search, while the search time of our schemes mainly depends on the results of the first screening. As the file sets grows, the result of the first screening increases as well, which makes the whole search time grow.

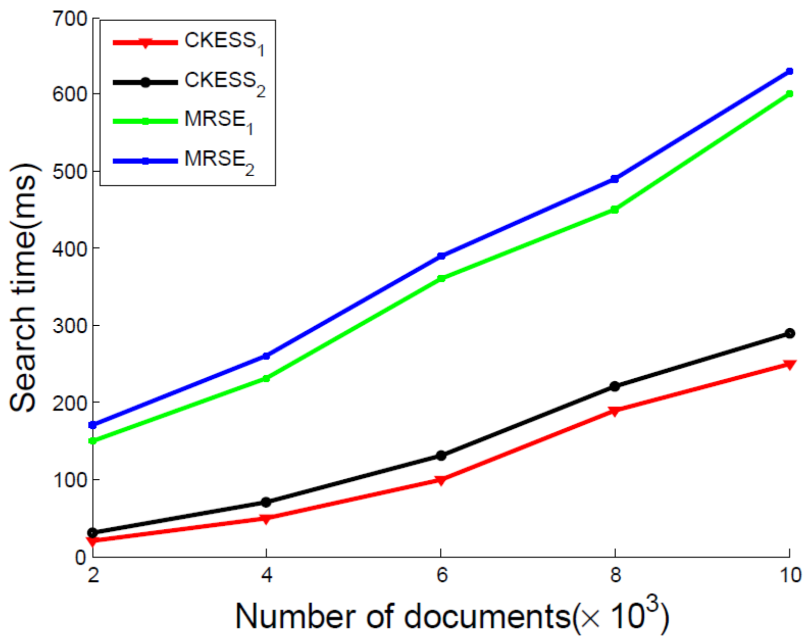

Figure 7: The time of search efficiency

7 Conclusion

In this paper, we propose privacy-preserving content-aware search by using the two-level index. The scheme makes the two-level index and expands the central word for improving efficiency and accuracy. At first, through the matching of the central extension word and the first layer index, the file set range is reduced. Then, the similarity calculation is performed on the file set and the trapdoor. Finally, the top-k file is returned to the user. Through experimental analysis, we verify our solution is more efficient, precise and secure than the existed.

Acknowledgement: This work is supported by the National Natural Science Foundation of China under grant U1836110, U1836208, U1536206, 61602253, 61672294; by the National Key R&D Program of China under grant 2018YFB1003205; by China Postdoctoral Science Foundation (2017M610574); by the Jiangsu Basic Research Programs-Natural Science Foundation under grant numbers BK20181407; by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund; by the Major Program of the National Social Science Fund of China (17ZDA092), Qing Lan Project; by the Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET) fund, China.

References

Cao, N.; Wang, C.; Li, M. (2014): Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, vol.

25, no. 1, pp. 222-233.

Chen, X.; Huang, X.; Li, J. (2015): New algorithms for secure outsourcing of large-scale systems of linear equations. *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 1, pp. 69-78.

Chuah, M.; Hu, W. (2011): Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data. *International Conference on Distributed Computing Systems Workshops*, pp. 273-281.

Ciaccia, P.; Patella, M.; Rabitti, F. (1997): Indexing metric spaces with m-tree. *PROC. Quinto Convegno Nazionale Sebd*, vol. 97, pp. 67-86.

Dan, B.; Crescenzo, G, D.; Ostrovsky, R. (2004): Public key encryption with keyword search. *Advances in Cryptology-Eurocrypt 2004*, pp. 506-522.

Fu, Z.; Ren, K.; Shu, J. (2016): Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546-2559.

Fu, Z.; Shu, J.; Wang, J. (2015): Privacy-preserving smart similarity search based on simhash over encrypted data in cloud computing. *Journal of Internet Technology*, vol. 16, no. 3, pp. 453-460.

Fu, Z.; Sun, X.; Liu, Q. (2015): Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. *IEICE Transactions on Communications*, vol. 98, no. 1, pp. 190-200.

Fu, Z.; Sun, X.; Linge, N. (2014): Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query. *IEEE Transactions on Consumer Electronics*, vol. 60, no. 1, pp. 164-172.

Fu, Z.; Wu, X.; Guan, C. (2016): Toward 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.

Fu, Z.; Wu, X., Wang, Q. (2017): 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.

Fu, Z.; Xia, L.; Sun, X. (2018): Semantic aware searching over encrypted data for cloud computing. *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2359-2371.

Gao, C. Z.; Cheng, Q.; He, P. (2018): Privacy-preserving naive bayes classifiers secure against the substitution-then-comparison attack. *Information Sciences*, vol. 444, pp.72-88.

Har-Peled, S.; Indyk, P.; Motwani, R. (1998): Approximate nearest neighbor: towards removing the curse of dimensionality. *ACM Symposium on Theory of Computing*, pp. 604-613.

Huang, F.; Fu, Z.; Sun, X. (2016): Privacy-preserving outsourced gene data search in encryption domain. *Security and Communication Networks*, vol. 9, no. 18, pp. 5178-5186.

Kurasawa, H.; Takasu, A.; Adachi, J. (2008): Huffman-DHT: index structure refinement scheme for P2P information retrieval. Information Retrieval. *International*

Symposium on Applications and the Internet, pp. 111-117.

Kuzu, M.; Islam, M, S.; Kantarcioglu, M. (2012): Efficient similarity search over encrypted data. *International Conference on Data Engineering*, pp. 1156-1167.

Leslie, H.; Jain, R.; Birdsall, D. (1995): Efficient search of multi-dimensional b-trees. *International Conference on Very Large Data Bases*, vol. 95, pp. 11-15.

Li, H.; Liu, D.; Dai, Y. (2018): Personalized search over encrypted data with efficient and secure updates in mobile clouds. *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 97-109.

Li, H.; Yang, Y.; Dai, Y. (2017): Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data. *IEEE Transactions on Cloud Computing*.

Li, H.; Yang, Y.; Luan, T, H. (2016): Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data. *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 312-325.

Li, J.; Chen, X.; Chow, S. S. M. (2018): Multi-authority fine-grained access control with accountability and its application in cloud. *Journal of Network & Computer Applications*, vol. 112, pp. 89-96.

Li, J.; Liu, Z.; Chen, X. (2015): L-EncDB: a lightweight framework for privacy-preserving data queries in cloud computing. *Knowledge-Based Systems*, vol. 79, pp. 18-26.

Li, J.; Wang, Q.; Wang, C. (2014): Fuzzy keyword search over encrypted data in cloud computing. *International Journal of Engineering Research and Applications*, vol. 4, pp. 441-445.

Li, R.; Liu, A, X.; Wang, A, L. (2016): Fast and scalable range query processing with strong privacy protection for cloud computing. *Transactions on Networking*, vol. 24, no. 4, pp. 2305-2318.

Liang, K.; Huang, X.; Guo, F. (2016): Privacy-preserving and regular language search over encrypted cloud data. *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2365-2376.

Lin, D. (1998): An information-theoretic definition of similarity. *International Conference on Machine Learning*, vol. 1, pp. 296-304.

Miller, G. (1990): Wordnet: an on-line lexical database. *International Journal of Lexicography*, vol. 3, no. 4, pp. 235-244.

Qiu, T.; Chen, N.; Li, K. (2018): How Can Heterogeneous Internet of Things Build our Future: A Survey. *IEEE Communications Surveys & Tutorials*.

Song, D, X.; Wagner, D.; Perrig, A. (2002): Practical techniques for searches on encrypted data. *Proceeding 2000 IEEE Symposium on Security and Privacy*, pp. 44-55.

Wang, B.; Li, M.; Wang, H. (2015): Circular range search on encrypted spatial data. *Communications and Network Security*, vol. 11, pp. 182-190.

Wang, J.; Yu, X.; Zhao, M. (2015): Privacy-preserving ranked multi-keyword fuzzy search on cloud encrypted data supporting range query. *Arabian Journal for Science and Engineering*, vol. 40, no. 8, pp. 2375-2388.

Weng, J.; Weng, J.; Zhang, Y. (2018): BENBI: scalable and dynamic access control on the northbound interface of SDN-based VANET. *IEEE Transactions on Vehicular Technology*.

Wong, W, K.; Cheung, D, W.; Kao, B. (2009): Secure KNN computation on encrypted databases. *ACM SIGMOD International Conference on Management of Data*, pp. 139-152.

Xu, W.; Li, H.; Dai, Y. (2019): Enabling efficient and geometric range query with access control over encrypted spatial data. *IEEE Transactions on Information Forensics and Security*, vol. 14, no.4, pp. 870-885.

Yang, J.; Liu, Z.; Li, J. (2014): Multi-key searchable encryption without random oracle. *International Conference on Intelligent Networking and Collaborative Systems*, pp. 79-84.

Yang, A.; Xu, J.; Weng, J. (2018): Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage. *IEEE Transactions on Cloud Computing*.

Zhang, S.; Li, H.; Dai, Y. (2018): Verifiable outsourcing computation for matrix multiplication with improved efficiency and applicability. *IEEE Internet of Things Journal*, vol. 5, no. 6.