Privacy-Preserving Smart Similarity Search Based on Simhash over Encrypted Data in Cloud Computing

Zhangjie Fu[1,2], Jiangang Shu[1,2], Jin Wang[1,2], Yuling Liu[2,3], Sungyoung Lee[4]
[1]School of Computer & Software, Nanjing University of Information Science & Technology, China
[2]Jiangsu Engineering Centre of Network Monitoring, Nanjing University of Information Science and Technology, China
[3]College of Computer Science and Electronic Engineering, Hunan University, China
[4]Department of Computer Engineering, Kyung Hee University, Korea
{wwwfzj, kennethshu}@126.com, wangjin@nuist.edu.cn, yuling_liu@126.com, sylee@oslab.khu.ac.kr

Abstract

In recent years, due to the appealing features of cloud computing, more and more sensitive or private information has been outsourced onto the cloud. Although cloud computing provides convenience, privacy and security of data becomes a big concern. For protecting data privacy, it is desirable for the data owner to outsource sensitive data in encrypted form rather than in plain text. However, encrypted storage will hinder our legal access, e.g., searching function. To deal with this dilemma, a considerable number of searchable encryption schemes have been proposed in this field. However, almost all of existing schemes focus on keyword-based query rather than document-based query, which is a crucial requirement for real world application. In this paper, we propose a similarity search method for encrypted document based on simhash. Through our scheme, data users can find similar encrypted documents stored in cloud by submitting a query document. In order to scale well for large data sources, we build a trie-based index to improve search efficiency in our solution. Through rigorous privacy analysis and experiment on real-world dataset, our scheme is secure and efficient.

Keywords: Similarity search, Document search, Searchable encryption, LSH, Cloud computing.

1 Introduction

In recent years, cloud computing has achieved great development due to the fact that it relieves the burden of data storage and data management. Hence, a large amount of data, ranging from emails to personal health records, is increasingly outsourced onto public clouds, such as Amazon Web Services [1], Microsoft Azure [2], Apple iCloud [3], Google AppEngine [4]. In the meantime, privacy and security of outsourced data has increasingly aroused our attention. For protecting data privacy, it is a recommended practice for the data owner to outsource sensitive data in encrypted form, which protects information privacy and alerts unauthorized access. However, data encryption makes data utilization (e.g., search operation) more difficult. The trivial solution to download the whole encrypted data firstly and then decrypt it locally is obviously impractical, due to the huge bandwidth and computation burden.

In fact, a considerable number of schemes based on searchable encryption [5] have been proposed to tackle the problem. Typically, almost all such schemes follow the model of keyword-based query, in which data users will retrieve relevant data files from the cloud by submitting several query words. In a real-world application, however, there usually exists a need to use a query document for searching similar ones. The alternative solution that users conclude and extract keywords manually from the query document firstly and then query these keywords is trivial. Moreover, the performance of this approach will be largely affected by extracted keywords, which depend on users in different cultural backgrounds. Apparently, using an entire document as query will be likely to get better retrieval results compared with the method to use just a few keywords as query, but the document-based approach is more complex and computationally demanding.

A document-based similarity search problem consists of a collection of documents that are characterized by some features, a query document and a similarity metric to measure the similarity between documents. This process is similar with image processing [25-26]. Currently, in document retrieval field, most of document representation schemes are based on vector space model, latent semantic indexing and other language models. Among them, vector space model (VSM) [27] is most widely adopted, which usually uses "TF × IDF" for term weighting and constructs a bag of words for feature description. TF (term frequency) is the occurrence of the term appearing in the document and IDF (inverse document frequency) is a function of the number of document where a term appears. A term weighted vector is constructed for each document and similarity between two documents can be measured by cosine distance. It is apparent that such method will take up much storage since each document is represented by a high dimensional vector. In addition, the computation of VSM when computing the similarity between documents is time consuming. Therefore, we need to find an efficient

*Corresponding author: Sungyoung Lee; E-mail: sylee@oslab.khu.ac.kr
DOI: 10.6138/JIT.2015.16.3.20140918

dimensionality reduction method while supporting similarity search over large data sources.

In this paper, the basic idea of our scheme is the state-of-art approximate near neighbor search algorithm in high dimension spaces called locality sensitive hashing (LSH) [17]. LSH is widely used for fast similarity search on plain data in information retrieval community (e.g., [18]). It projects high-dimensional objects to compact binary codes called fingerprints and makes similar fingerprints for similar objects. In our scheme, we use the simhash algorithm [19], a kind of LSH, to map the features of each document into a fixed-length fingerprint and thereby the similarity between documents can be measured by calculating the hamming distance between fingerprints. Moreover, a trie-based index covering fingerprints is constructed to improve search efficiency. Rigorous privacy analysis shows that our scheme is secure. Through experiment on real-world dataset and performance analysis, our scheme is quite efficient and infeasible.

The rest of paper is organized as follows. We discuss the background and related work on searchable encryption and LSH in Section 2. Section 3 shows the detailed design of similarity search scheme. Sections 4 and 5 present the efficiency improvement and performance analysis, respectively. Finally, we conclude the paper in Section 6.

2 Problem Formulation

2.1 Searchable Encryption

Searchable encryption is a new developing information security technique and it enables users to search over encrypted data through keywords without having to decrypt it at first. It is recommended to build a searchable index in most of existing searchable encryption schemes. The searchable index is encrypted in such a way that the cloud server cannot deduce the plaintext from the index, while allowing the search operation from authorized users. The first practical searchable encryption scheme is proposed by Song et al. [5], in which each word is encrypted independently under a special construction. Later on, some security definitions and improvement schemes based on similar index have been proposed by Goh [6], Chang et al. [7] and Curtmola et al. [8].

In recent years, a large number of searchable encryption [9-15] schemes in cloud computing have been proposed. Ownership in Deduplicated Cloud Storage is discussed [24]. In such schemes, many interesting and important issues have been proposed and discussed.

Secure Ranked Search: Wang et al. [9] propose a secure ranked search scheme based on "TF×IDF." Without knowledge of specific keyword weight, the cloud server can also help rank relevant data files using order preserving symmetric encryption. But this scheme only supports single keyword search. Then Cao et al. [10] propose a privacy-preserving ranked search scheme supporting multi-keyword, which uses secure KNN method [16] to calculate the similarity based on VSM. Based on [10], in order to solve the top- problem, Sun et al. [11] adopt a MDB-tree to construct the searchable index, thereby improving search efficiency.

Fuzzy Search and Similarity Search: Li et al. [12] firstly propose a fuzzy keyword search scheme over encrypted cloud data, which combines edit distance with wildcard-based technique to construct fuzzy keyword sets. Although this fuzzy search scheme addresses problems of minor typos and format inconsistence to some extent, its performance is dependent on parameter d of edit distance. Kuzu et al. [13] propose a similarity search scheme which uses minhash based on Jaccard distance to support fault tolerant keyword search.

Semantic Search and Preferred Search: Fu et al. [14] propose a semantic keyword search scheme based on stemming algorithm, which helps users find relevant documents containing semantically close keywords related to the query word. Shen et al. [15] propose a preferred keyword search scheme, in which the search result will faithfully respect the user's preference. But how to measure keyword preference is ignored in that paper.

2.2 LSH

LSH is a special kind of hashing method, which uses a set of hash functions to hash codes of similar objects collide with high probability and while dissimilar ones not, such that for objects A and B:

$$Pr[h(A) = h(B)] = sim(A, B) \tag{1}$$

Where $sim(A, B) \in [0,1]$ is a specific similarity metric. Manku et al. [20] represent a detailed algorithm for simhash with Hamming distance. It maps high-dimensional vectors to small-sized fingerprints. The input of simhash algorithm is an -dimension vector, while the output is an m-bit fingerprint ($m < n$). Simhash has two main features: (1) The fingerprint of a data object is the simhash value of the features of the data object; (2) Similar data objects usually have similar fingerprints. Since the size of the fingerprint is fixed and relatively small, it will not cost much storage. These features enable simhash to be suitable for document similarity search.

3 Design

3.1 System Architecture Overview

As depicted in Figure 1, a complete search system

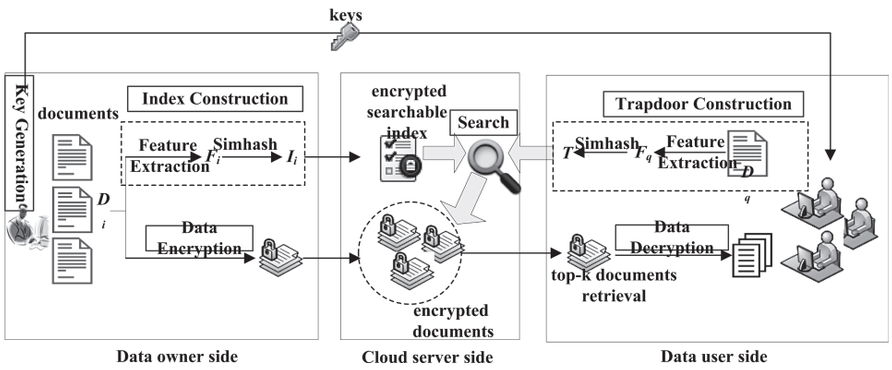


Figure 1 Architecture of Similarity Search System

consists of three different entities: the data owner, the cloud server, the data user. Given a huge size of document collection D, the data owner firstly extracts the features F_i of each document and generates the fingerprint I_i based on F_i. Then the owner integrates each I_i as the encrypted searchable index I_i and outsources them, together with the encrypted form C of document collection D, to the cloud server. Authorized users delegated by the data owner can make use of search service offered by the cloud server. To retrieve similar documents of a query document, the data user only needs to submit a search request T to the cloud. Upon receiving a search request from the authorized user, the cloud server will conduct designated search operation over the index and send back relevant encrypted documents, which have been well ranked according to some ranking criteria (e.g., Hamming distance). At last, the data user can decrypt the returned documents using the keys distributed by the data owner. To reduce communication cost, the data user can send the search request with an optional parameter k so that he just needs to retrieve top-k documents that are most relevant to the search request. The distribution of keys is out scope of our paper.

3.2 Notations

For the sake of clarity, we introduce the main notations used in this paper.

- D -- the plaintext document collection, denoted as $D = \{D_1, ..., D_m\}$.
- C -- the encrypted document collection, denoted as $C = \{C_1, ..., C_m\}$.
- $F_i = \{f_{i_1}, ..., f_{i_z}\}$ -- the set of features characterized D_i.
- $I = \{I_1, ..., I_m\}$ -- the encrypted searchable index for documents, I_i is the fingerprint of F_i.
- T -- the trapdoor of the query document D_q.
- π -- a hash function, $\{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^h$.
- sk -- a symmetric key, $sk \xleftarrow{R} \{0,1\}^k$.
- $Enc(sk, \cdot)$ -- the symmetric encryption algorithm.
- $Dec(sk, \cdot)$ -- the symmetric decryption algorithm.
- $R_{T,k}$ -- the ranked documents according by the similarity score with T.

3.3 Basic Search Scheme

The similarity search system involves the following algorithms:

Key Generation: In this initialization phase, the data owner produces a symmetric key $sk \xleftarrow{R} \{0,1\}^k$.

Index Construction: Given the document collection D, the data owner extracts the features F_i of each document D_i in D. And then he generates the fingerprint I_i of F_i using simhash algorithm. After that, the searchable index $I = \{I_1, ..., I_m\}$ is generated.

Data Encryption: Given the document collection D, the data owner encrypts each document D_i using $Enc(sk, \cdot)$ algorighm. After that, he sends encrypted document collection C along with searchable index I to the cloud server.

Trapdoor Construction: For a given query document D_q, the data user needs to extract its features and generates its fingerprint (trapdoor T) as same as index construction. After that, the user sends the trapdoor T with an optional parameter k to the cloud.

Search: With the trapdoor T, the cloud server conducts designated search operation over the index I, and returns top k encrypted documents $R_{T,k}$ sorted by the similarity score with T.

Data Decryption: Once the encrypted documents $R_{T,k}$ are retrieved, the user decrypts them with $Dec(sk, \cdot)$ algorithm to obtain their plain version.

3.4. Details of Main Steps

3.4.1 Feature Extraction

Since keywords are practical tools to summarize the content of document, we decide to extract the keywords of the document as its features. In order to obtain accurate keywords, we adopt the most widely measurement "TF ×

IDF," where TF represents its significance within a document and IDF indicates the degree of distinction within a whole document collection. We can calculate TF × IDF of each word within a document D_i as formula 2, sort them in descending order and pick up top z words as the features of the corresponding document.

$$S_{w,D_i} = \ln(1 + f_{d,w}) \times \ln(1 + \frac{m}{f_w}) \quad (2)$$

Where $f_{d,w}$ is the TF of the word in document D_i, f_w is the number of documents containing the word w and m is the size of the whole document collection. Therefore, the document D_i is characterized by features $F_i = \{f_{i_1}, ..., f_{i_z}\}$ where f_{i_j} is the word extracted.

3.4.2 Simhash

Given the features F_i (F_q) of a document D_i (D_q), the data owner or the data user will conduct simhash algorithm to get the fingerprint of F_i (F_q). Figure 2 shows the process of simhash, which can be also described as follows:

(1) Initialize a h-dimension vector V where each dimension is set as 0;
(2) Each feature f_{i_j} in F_i (F_q) is projected into a h-bit signature using the traditional hash function π. If the i-th bit of this signature is 1, the i-th dimension of V plus 1. Otherwise, it will minus 1;
(3) At last, generate a h-bit simhash fingerprint I_i (T) according to every dimension of vector V. If the i-th dimension of V is positive number, and then the i-th bit of I_i (T) is 1. Otherwise, it will be 0.

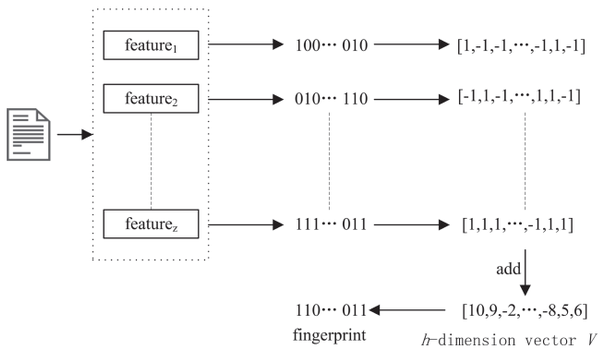


Figure 2 Process of Simhash

3.4.3 Search

After the searchable index I is built well, given the query trapdoor T based on D_q, the cloud server can perform similarity search over the index I. Since each sub-index I_i and the trapdoor T are both the fingerprint of simhash actually, the similarity of two documents can be calculated with hamming distance as follows:

$$sim(D_i, D_q) = H(I_i, T) \quad (3)$$

$H(S_1, S_2)$ is hamming distance between two strings (S_1 and S_2) of equal length. It is calculated by counting the number of positions where the corresponding symbols are different. For example, $H(1011101, 1001001 = 2)$. The smaller hamming distance means that two documents are more similar. After calculating the similarity of each sub-index I_i with T, the cloud server ranks all similarity scores in ascending order and selects top k documents as $R_{T,k}$.

3.5 Security Analysis

Document Privacy: All outsourced documents are cipher under a symmetric encryption algorithm and their privacy can be guaranteed. No information will be leaked except their length and document IDs.

Index Confidentiality and Trapdoor Confidentiality: During both index construction and trapdoor construction, simhash algorithm is adopted in this paper. Each feature in a document is firstly mapped into a h-bit signature, which is protected by a traditional one-way hash function π. Each sub-index I_i or the trapdoor T is a binary string of equal length, which is produced by aggregating all signatures of features in one document. Therefore, each sub-index I_i or the trapdoor T is an irregular binary string and the cloud server cannot deduce any useful information from them.

4 Efficiency Improvement

In the last section, we present a solution which requires the trapdoor T to calculate the similarity score with each document in cloud. In order to scale well for large data sources, we adopt to build a trie-based index to improve search efficiency, as shown in Figure 3. The key idea behind the trie is that all the descendants of a node have a common prefix associated with that node. In the trie-based index, each path from the root node to a leaf node represents a sub-index I_i. In the top-k search procedure, it adopts a depth-first method to start from the root node. In the search process downward, we should start from the root node and read the characters in the query in sequence. For each

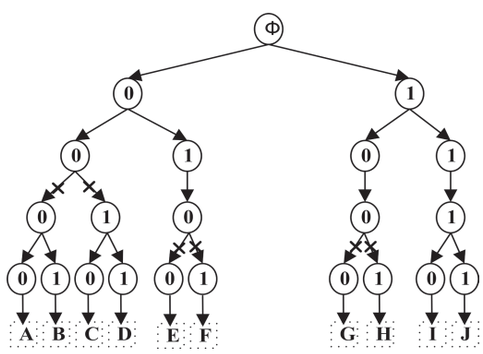


Figure 3 Example of Trie-Based Index

read character, it always chooses the node with the unused minimum hamming distance, predicting the minimum possible final score to be obtained. If the predicted final score is larger than the maximum score among the top-k documents chosen, search process will return to its parent, otherwise, it will go down to the child node at the next level. The search process is performed recursively until the top-k documents are produced. The details can be seen Algorithm 1. Through this search method, search process doesn't need to go through the whole trie and only accesses a part of nodes actually, which saves much time. Figure 3 shows an example: Given a trapdoor T = "1111,", when k = 2, only the nodes, J and I, are returned to the user. The cross sign means search process doesn't need to go through the path.

The following notations are used in the pseudo code for our search algorithm:

- x -- a document, it can be defined by a tree path ($I_{x,1}$, $I_{x,2}$, ..., $I_{x,h}$).
- T_i -- the -th symbol in the trapdoor T.
- S_i -- the hamming distance that is equal to $H(I_{x,i}, T_i) = I_{x,i} xor T_i$.
- $score(\cdot)$ -- the current score of a path or document, e.g., $score(\cdot) = \sum_{i=1}^{h} S_i$.
- L_k -- the list for select top-k documents that are stored in ascending order according to $score(x)$.
- M_k -- the score of the k-th document in L_k.
- $I.Root$ -- the root node of the trie index I.
- p -- the node in I.
- $p.child[T_i]$ -- the child node of p that is equal to T_i.
- $p.child[another]$ -- the child node of p not equaling T_i.

Algorithm 1 Search (trie I, trapdoor T, k)

```
begin
findTopK ( I.Root,T, 0, 0, k);
return L_k;
end

procedure findTopK (node p, trapdoor T, int i, int score,
int k);
if (p is leafnode) then
   if (|L_k| < k) then
      insert document x into L_k according to score(x);
    else
      if (score < M_k) then
         delete k-th document from L_k;
         insert document x into L_k according to score(x);
      end if
   end if
else
   if (T_i is a child node in p) then
      if (|L_k| < k or score < M_k) then
      score = score + H(p.child[T_i], T_i);
      findTopK (p.child[T_i], T, i + 1, score, k);
   end if
end if
if (there is another child in p) then
      score = score + H(p.child[another], T_i);
      if (|L_k| < k or score < M_k) then
         findTopK (p.child[another], T, i + 1, score, k);
      end if
   end if
end if
end procedure
```

5 Performance Evaluation

To evaluate the performance of our proposed search scheme, we conduct a thorough experiment on a 2.83 GHZ Intel Core (TM) processor, Windows 7 operating system with a RAM of 4G. We use Newsgroup 20 [21] for experiment in our scheme. Newsgroup 20 is a publicly available real-world dataset, which contains approximately 20,000 newsgroup documents. It has been widely used in text applications of machine learning techniques, such as text classification and text clustering. In our experiment, we randomly choose 6,000 documents as our final experiment corpus.

5.1 Experimental Setup

5.1.1 Feature Extraction

In order to extract the accurate keywords as features, we need to process the raw texts in each document through the following steps: (1) word splitting; (2) filtering stop words like "a," "the," "such," "from," etc.; (3) converting each word into lowercase; (4) getting the root of each word (stemming). Here we use Porter Stemming Algorithm [22] as our stemming algorithm. After that, we compute the TF × IDF of each word as formula 2 and only pick up top 10 words in one document as keywords.

5.1.2 Simhash

For the features extracted in each document, we need to hash them to a fingerprint. In our experiment, we choose SHA-1 as our hash function π. So each document will be hashed to a 128-bit fingerprint.

5.2 Efficiency

To the best of our knowledge, parameters of precision and recall are usually used to measure the performance in information retrieval field. The precision and recall in our search scheme are mainly decided upon the performance

of simhash, which has been widely discussed [19][23]. Therefore, the performance in this section mainly focuses on the efficiency of algorithms: index construction, trapdoor construction and search.

5.2.1 Index Construction

In our experiment, since the adoption of trie-based index, the index construction contains three steps: feature extraction, simhash and trie-based index building. Figure 4 shows the time cost of index construction is nearly linear with the number of documents. Given 6,000 documents, it takes about 35 minutes to build up the whole index. In the index construction, the step of feature extraction, as the major computation, takes up most of the time. The other two steps (simhash and trie-based index building) are quite efficient. Figure 5 shows the time of the step of building trie-based index is also nearly linear with the number of documents. Compared with feature extraction, the time of building trie-based index is quite efficient and has little impact on the time of the whole index construction. There is no denying that index construction is a burden for the data owner but it is kept within the bearable limit. In addition, it is just a one-time cost.

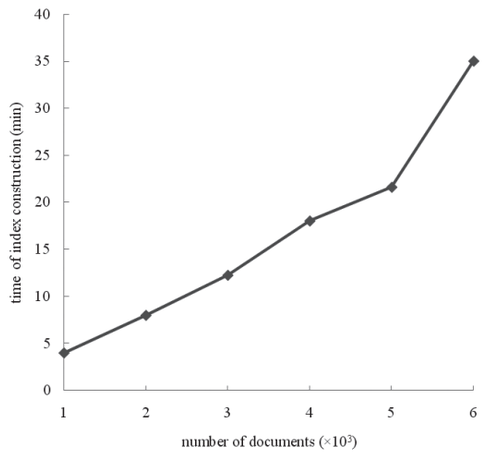


Figure 4 Time of Index Construction

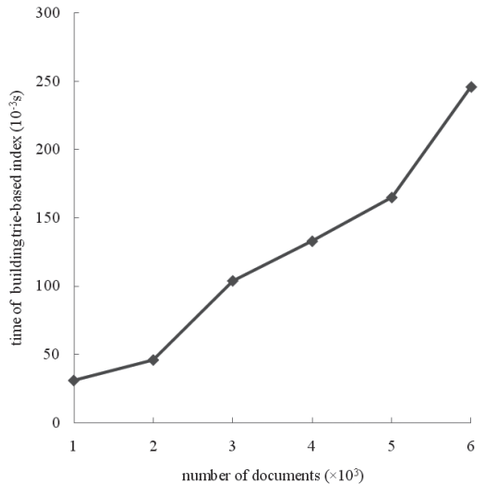


Figure 5 Time of the Step of Building Trie-Based Index

5.2.2 Trapdoor Construction

In the trapdoor construction, the data user needs to extract the features of the query document at first and then hashes them to a fingerprint using simhash. Figure 6 shows the time cost to generate the trapdoor for different size of document. We test a large number of documents with size varying from 1 KB to 60 KB and the time of trapdoor construction is not more than 1 second. Therefore, this process will not impose a burden on the data user.

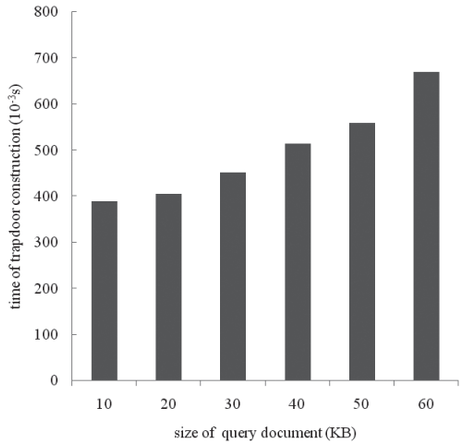


Figure 6 Time of Trapdoor Construction

5.2.3 Search

The search process conducted at the cloud server consists of computing and ranking the similarity scores of relevant documents. The search algorithm will terminate at once if the top-k documents are selected. Figure 7 shows the time for trie-based search, compared with baseline search with respect to the number of documents. In the baseline search, the cloud server needs to calculate the similarity score of each document stored in cloud. Therefore, the time of baseline search rises with the number of documents. Compared with baseline search, our trie-based search algorithm is quite efficient and barely affected by the number of documents.

6 Conclusion

In this paper, we propose a document-based similarity search scheme over encrypted cloud document. In our solution, each document is transformed into a fingerprint with simhash and hamming distance is used as the similarity score between documents. Moreover, trie-based index is adopted to address the top-k problem and search

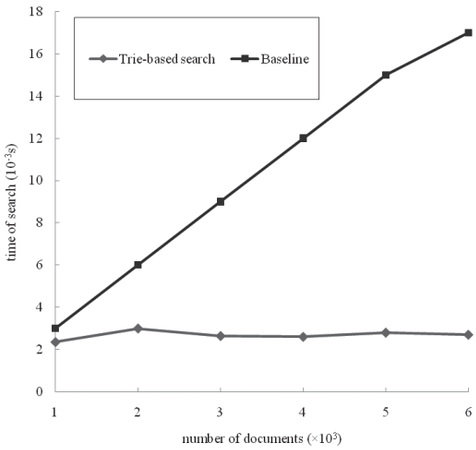


Figure 7 Time of Search

efficiency is improved greatly. Experiment on real-world dataset proves our scheme is quite efficient.

Acknowledgements

This work is supported by the NSFC (61373133, 61232016, 61173141, 61173142, 61173136, 61103215, 61373132, 61402234, 61272421), GYHY201206033, 201301030, 2013DFG12860, BC2013012, PAPD fund, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Hunan province science and technology plan project fund (2012GK3120), the Scientific Research Fund of Hunan Provincial Education Department (10C0944), the Prospective Research Project on Future Networks of Jiangsu Future Networks Innovation Institute (BY2013095-4-10), Jiangsu Province Natural Science Research Program (BK2012461), and CSC fund.

References

[1] Amazon Web Services, 2015, http://aws.amazon.com

[2] Microsoft Azure, 2015, http://www.windowsazure.com.

[3] Apple iCloud, 2015, https://www.icloud.com/

[4] Google AppEngine, 2015, https://appengine.google.com/

[5] Dawn Xiaodong Song, David Wagner and Adrian Perrig, *Practical Techniques for Searches on Encrypted Data*, *Proc. of IEEE Symposium on Security and Privacy*, Berkeley, CA, May, 2000, pp.44-55.

[6] Eu-Jin Goh, *Secure Indexes*, 2004, https://eprint.iacr.org/2003/216.pdf

[7] Yan-Cheng Chang and Michael Mitzenmacher, *Privacy Preserving Keyword Searches on Remote Encrypted Data*, *Proc. of the 3th International Conference on Applied Cryptography and Network Security*, New York, June, 2005, pp.442-455.

[8] Reza Curtmola, Juan Garay, Seny Kamara and Rafail Ostrovsky, *Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions*, *Proc. of the 13th ACM Conference on Computer and communications security*, Alexandria, October, 2006, pp.79-88.

[9] Wang Cong, Cao Ning, Li Jin, Ren Kui and Lou Wenjing, Secure ranked keyword search over encrypted cloud data, *Proc. of the 30th IEEE International Conference on Distributed Computing System*, Genoa, Italy, June, 2010, pp.253-262.

[10] Ning Cao, Cong Wang, Li Ming, Kui Ren and Wenjing Lou, *Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data*, *IEEE Transactions on Parallel and Distributed Systems*, Vol.25, No.1, 2014, pp.222-233.

[11] Wenhai Sun, Bing Wang, Ning Cao, Ming Li, Wenjing Lou, Y. Thomas Hou and Hui Li, *Privacy-Preserving Multi-keyword Text Search in the Cloud Supporting Similarity-Based Ranking*, *Proc. of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, Hangzhou, China, May, 2013, pp.71-82.

[12] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren and Wenjing Lou, *Fuzzy Keyword Search over Encrypted Data in Cloud Computing*, *Proc. of IEEE INFOCOM 2010*, San Diego, CA, March, 2010, pp.1-5.

[13] Mehmet Kuzu, Mohammad Saiful Islam and Murat Kantarcioglu, *Efficient Similarity Search over Encrypted Data*, *Proc. of the 28th IEEE International Conference on Data Engineering*, Washington, DC, April, 2012, pp.1156-1167.

[14] Zhangjie Fu, Jiangang Shu, Xingming Sun and Daxing Zhang, *Semantic Keyword Search Based on Trie over Encrypted Cloud Data*, *Proc. of the 2nd International Workshop on Security in Cloud Computing*, Kyoto, Japan, June, 2014, pp.59-62.

[15] Zhirong Shen, Jiwu Shu and Wei Xue, *Preferred Keyword Search over Encrypted Data in Cloud Computing*, *Proc. of the 21st IEEE/ACM International Symposium on Quality of Service*, Montreal, Canada, June, 2013, pp.1-6.

[16] Wai Kit Wong, David Wai-Lok Cheung, Ben Kao and Nikos Mamoulis, *Secure kNN Computation on Encrypted Databases*, *Proc. of the 2009 ACM SIGMOD International Conference on Management of Data*, Providence, RI, June, 2009, pp.139-152.

[17] Piotr Indyk and Rajeev Motwani, *Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality*, *Proc. of the Thirtieth Annual ACM Symposium On Theory of Computing*, Dallas, TX, May, 1998, pp.604-613.

[18] Qin Lv, William Josephsona, Zhe Wang, Moses Charikar and Kai Li, *Multi-probe LSH: Efficient Indexing for High-Dimensional Similarity Search*, *Proc. of the 33rd International Conference on Very Large Data Bases*, Vienna, Austria, September, 2007, pp.950-961.

[19] Moses S. Charikar, *Similarity Estimation Techniques from Rounding Algorithms*, *Proc. of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, Montreal, Canada, May, 2002, pp.380-388.

[20] Gurmeet Singh Manku, Arvind Jain and Anish Das Sarma, *A Detecting Near-Duplicates for Web Crawling*, *Proc. of the 16th International Conference on World Wide Web*, Banff, Canada, May, 2007, pp.141-150.

[21] *The 20 newsgroups data set*, 2014, http://qwone.com/~jason/20Newsgroups/

[22] Martin F. Porter, *An Algorithm for Suffix Stripping*, *Program*, Vol.14, No.3, 1980, pp.130-137.

[23] Jiaming Xu, Pengcheng Liu, Gaowei Wu, Zhengya Sun, Bo Xu and Hongwei Hao, *A Fast Matching Method Based on Semantic Similarity for Short Texts*, *Proc. of NLPCC 2013*, Chongqing, China, November, 2013, pp.299-309.

[24] Chia-Mu Yu, Chi-Yuan Chen and Han-Chieh Chao, *Proof of Ownership in Deduplicated Cloud Storage with Mobile Device Efficiency*, *IEEE Network*, Vol.29, No.2, 2015, pp.51-55.

[25] Jian Li, Xiaolong Li, Bin Yang and Xingming Sun, *Segmentation-Based Image Copy-Move Forgery Detection Scheme*, *IEEE Transactions on Information Forensics and Security*, Vol.10, No.3, 2015, pp.507-518.

[26] Hui Zhang, Q. M. Jonathan Wu, Thanh Minh Nguyen and Xingmin Sun, *Synthetic Aperture Radar Image Segmentation by Modified Student's t-Mixture Model*, *IEEE Transaction on Geoscience and Remote Sensing*, Vol.52, No.7, 2014, pp.4391-4403.

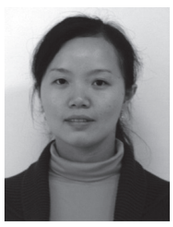
[27] Bin Gu and Victor S. Sheng, *Feasibility and Finite Convergence Analysis for Accurate On-Line v-Support Vector Learning*, *IEEE Transactions on Neural Networks and Learning Systems*, Vol.24, No.8, 2013, pp.1304-1315.

Biographies

Zhangjie Fu obtained his PhD in computer science from the College of Computer, Hunan University, China, in 2012. Currently, he works as an Assistant Professor in College of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include network and information security, copyright protection technology.

Jiangang Shu received his BE in Network Technology and Engineering from Nanjing University of Information Science & Technology (NUIST), China, in 2012. He is currently pursuing his MS in computer science and technology in the same university. His research interests include cloud security, steganography, and network and information security.

Jin Wang received PhD degree from Kyung Hee University Korea in 2010. Now, he is a Professor in the Nanjing University of Information Science and technology. His research interests mainly include routing protocol and algorithm design, performance evaluation for wireless sensor networks.



Yuling Liu obtained her PhD in computer science from the College of Computer, Hunan University, China, in 2008. Currently, she works as an Assistant Professor in College of Computer Science and Electronic Engineering, Hunan University, China. Her research interests include network and information security, steganography.

Sungyoung Lee is a Professor in the Department of Computer Engineering, Kyung Hee University, Korea since 1993. His current research focuses on Ubiquitous Computing and Applications, Wireless Ad-hoc and Sensor Networks, Context-Aware Middleware, Sensor Operating Systems, Real-Time Systems and Embedded Systems.