International Journal of Grid Distribution Computing
Vol.7, No.5 (2014), pp.65-76
http://dx.doi.org/10.14257/ijgdc.2014.7.5.06

Efficient Keyword Search Scheme in Encrypted Cloud Computing Environment

Jiangang Shu, Xingming Sun, Lu Zhou and Jin Wang

School of Computer and Software & Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing 210044, China

Abstract

With the increasing popularity of cloud computing, more and more sensitive or private information has been outsourced onto the cloud server. For protecting data privacy, sensitive data usually has to be encrypted before outsourcing, which makes traditional search techniques based on plaintext useless. In response to the search over encrypted data, searchable encryption is a good solution in Information Security. However, most of existing searchable encryption schemes only support exact keyword search. That means they don't support searching for different variants of the query word, which is a significant drawback and greatly affects data usability and user experience. Recently, a fuzzy keyword search scheme proposed by some researchers aims at addressing the problems of minor typos and format inconsistence but couldn't solve the problem above. In this paper, we formalize the problem of semantic keyword-based search over encrypted cloud data while preserving privacy. Semantic keyword-based search will greatly improves the user experience by returning all the documents containing semantically close keywords related to the query word. In our solution, we use the stemming algorithm to construct stem set, which reduces the dimension of index. And the symbol-based trie is also adopted in index construction to improve the search efficiency. Through rigorous privacy analysis and experiment on real dataset, our scheme is secure and efficient.

Keywords: *Semantic search, searchable encryption, stemming algorithm, cloud computing*

1. Introduction

With the advent of cloud computing, more and more sensitive or private information has been outsourced onto the cloud in order to benefit from unlimited storage space and enormous computation power. However, data owners have no direct access on the outsourced data, thereby making the data at risk. The data stored in the cloud may suffer from malicious use or unauthorized access by the cloud service provider, classically considered as "honest-but-curious". A classic solution is that data owners encrypt their data before outsourcing onto the cloud. This solution protects the data but makes data utilization 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.

One popular approach to solve this dilemma is searchable encryption, which can retrieve specific files through keyword-based search with data protection and keyword privacy-preserving. In recent years, searchable encryption has attracted a lot of attention from a growing number of researchers. However, most of existing searchable

ISSN: 2005-4262 IJGDC

encryption schemes only support exact keyword search, except the fuzzy keyword search scheme proposed by Li et al. [11]. That is, it is common that the user's searching input might not exactly match those pre-set keywords due to different variants of the word with semantically close meaning, such as "compute", "computed", "computing", *etc.*, The native way to support semantically close keyword search is to generate all the keywords within proximity in a semantic sense from the original keyword and send all the semantically close keywords as queries to the cloud server. This basic solution achieves the basic goal of semantically close keyword search however it is ineffective due to the following reasons: on one hand, it costs user extra computation effort to seek out all the semantically close keywords; on the other hand, it requires several search operations in the server side where the complexity of search is proportional to the number of semantically close keywords. All these drawbacks signify the important need for new techniques that support searching flexibility, tolerating different variants of the query word.

In this paper, we formalize the problem of semantic keyword-based search over encrypted cloud data while preserving privacy. Moreover, we combine the searchable encryption technique with a stemming algorithm. This method eliminates the need of enumerating all the semantically close keywords both in index construction and query generation, and also reduces the dimension of the index. In addition, we propose an efficient semantic keyword-based search scheme, which greatly enhances searching flexibility by returning all the documents containing the semantically close keywords related to the given query word. Rigorous privacy analysis shows that our scheme is secure. Through experiments on the real-world dataset and performance analysis, we show our scheme is quite efficient and infeasible.

The rest of paper is organized as follows. We discuss the related work on searchable encryption in Section 2. In Section 3, we introduce the system model, threat model and our design goals and then briefly describe some notations and background knowledge used in our paper. Section 4 shows the detailed construction of semantic keyword-based search scheme. Section 5 presents performance analysis and Section 6 concludes the paper finally.

2. Related Work

Searchable encryption is a new information security technique and it can enable users search over encrypted outsourced data through keywords without decrypting the data at first. It is a common practice to build a searchable index in the most of existing searchable encryption schemes. The searchable index is encrypted to protect the cloud server from deducing the plaintext from the index, while allowing the search operation from authorized users. The first practical searchable encryption scheme in symmetric setting is proposed by Song, *et al.*, [1], in which keywords are encrypted by a deterministic encryption algorithm under a two-layer construction and users have to go through the whole document collection to search a certain keyword. After that, Goh [2], Chang, *et al.*, [3] and Curtmola, *et al.*, [4] propose the improvement schemes based on similar index to improve search efficiency. Boneh, *et al.*, [5] propose the first public key-based searchable encryption scheme, where anyone owning the private key can search the data encrypted by the public key. In recent years, many schemes and security definitions have been put forward under different applications [6-20]. Among those schemes, some interesting issues such as conjunctive or range query [6-9], multi-keyword search [10, 15, 17], result secure ranking [13, 14, 17], fuzzy keyword search

[11, 12, 15], similarity search [16] and preferred search [18] have been proposed and discussed. Specially, in order to address the problems of minor typos and format inconsistence, Li, *et al.*, [11] propose a fuzzy keyword search scheme which combines edit distance with wildcard-based technique to construct fuzzy set. This scheme might address the problem of different variants of a word to a certain extent, but the performance of the scheme is largely affected by the variable d of edit distance and the dimension of index also increases a lot.

3. Problem Formulation

3.1. System Model

In this paper, we consider a complete system model in cloud computing should involve the data owner, the data user and the cloud server, as shown Figure 1. In this model, in order to outsource the data onto the cloud, the data owner needs to build the searchable index, which is based on the stem set extracted from the keyword set. And then he outsources the encrypted documents C, together with the index, onto the cloud. Given a predefined keyword set W, authorized users can make use of search service on encrypted documents C provided by the cloud server. We assume the authorization between the data owner and the data user is appropriately done. In order to retrieve documents of his interest, the authorized user submits a search request to the cloud server. 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.

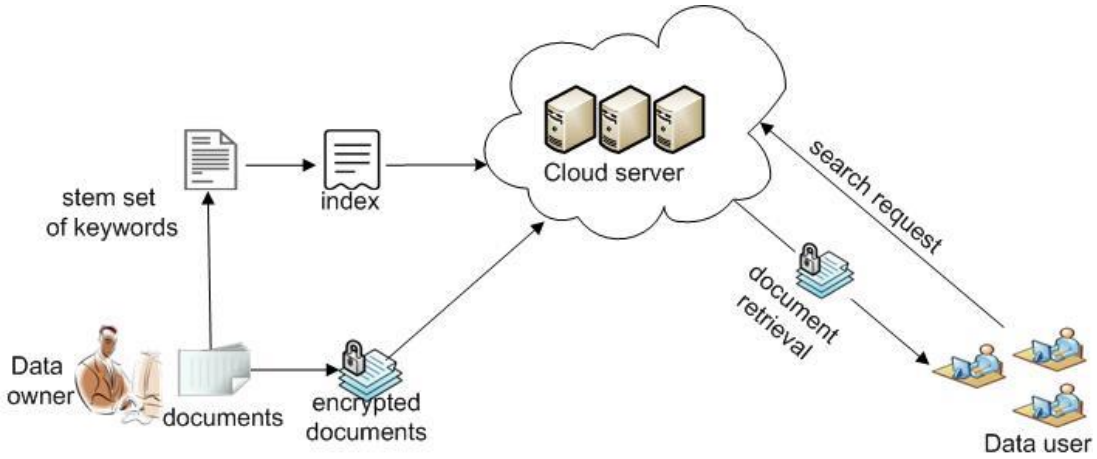

Figure 1. Architecture of Semantic Keyword-based Search

3.2. Threat Model

In our scheme, the cloud server is considered as “honest-but-curious” like many previous works [10, 17] of searchable encryption. That means, the cloud server will execute designated protocol honestly and correctly, but it is eager to get some sensitive information by inferring and analyzing data or index in its storage and the search request received during the protocol.

3.3. Design Goals

In this paper, we address the problem of semantic keyword-based search service over the encrypted cloud data. Specifically, we focus on the following goals:

- **Flexible Search** Our scheme should provide the flexible search service, which can return all the documents containing the semantically close keywords related a given query keyword.
- **Privacy-Preserving** The general goal is to protect the cloud server from learning additional sensitive information from document collection, index and search request. Specifically, we are concerned with document privacy, index privacy and query confidentiality.
- **Efficiency** The proposed schemes should be achieved with low communication and computation both on the data owner and user.

3.4. Notations

The main notations used in this paper are showed as below:

- D – the plaintext document collection, denoted as $D = \{D_1, \cdots, D_m\}$
- C – the encrypted document collection, denoted as $C = \{C_1, C_2, \cdots, C_m\}$.
- W – the keyword set consisting of n keywords, denoted as $W = \{w_1, w_2, \cdots, w_n\}$.
- sk – *a random key*, $sk \xleftarrow{R} \{0,1\}^k$.
- $f(sk, \cdot)$ – a one-way function, $\{0,1\}^* \times \{0,1\}^{sk} \rightarrow \{0,1\}^L$.
- S – the stem set of W, denoted as $S = \{s_1, \cdots, s_r\}$, $|S| < |W|$.
- $W(s_i)$ – the sub-set of keywords and their stem is s_i, $W(s_i) \subset W$.
- T_{s_i} – the trapdoor of the stem s_i, $T_{s_i} = f(sk, s_i)$.
- $\Delta = \{\alpha_i\}$ – the pre-defined symbol set, α_i is θ-bit binary vector, $|\Delta| = 2^\theta$;
- G – a symbol-based trie based on Δ.
- FID_{w_i} – the set of identifiers of files containing the keyword w_i.
- $Enc(sk, \cdot)$ – the symmetric encryption algorithm.
- $Dec(sk, \cdot)$ – the symmetric decryption algorithm.

3.5. Preliminaries

Stemming Algorithm: Stemming is a technique that aims at connecting morphological variants of a word together, which can reduce the inflected or derived words to their root form. It is widely adopted in Information Retrieval systems to improve performance. Note that although the purpose of stemming algorithm is to find the stem of the word, the stem doesn't need to be always meaningful. Indeed, the stem may have an incorrect form with respect to some lexical rules, *e.g.*, for the word "services" may have a root form "servic". In our semantic keyword-based search scheme, the stemming algorithm is performed for both the data owner and the data user, and the incorrectness of the stem has no impact on returned results or search performance.

Trie: Trie is also called prefix tree or radix tree, which is an $|\varepsilon|$-ary tree to store a set of words from an symbol set ε. The key idea behind is that all the descendants of a node have a common prefix associated with that node, as shown in Figure 2. To perform a search for a query word in the trie, we should start from the root node and read the characters in the query word in sequence. For each read character, the pointer will move to corresponding node in the trie from top to bottom. If such a node does not exist, the search operation will terminate immediately and return a failure result; when all

characters are read, the pointer arrives at a leaf node. That means, the query word belong to the trie and the search will return a success result.

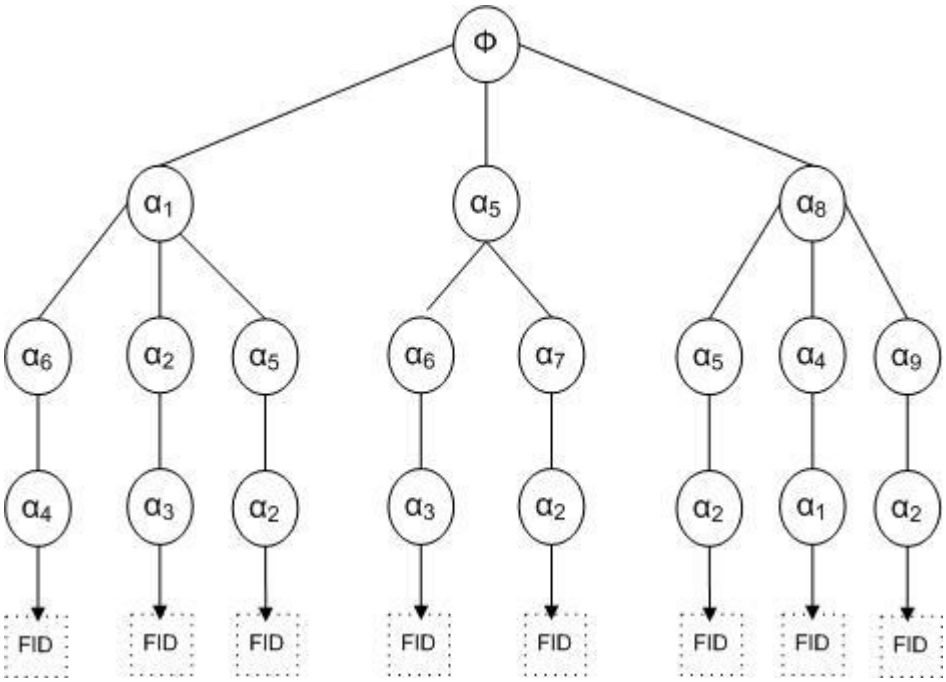

Figure 2. An Example of Trie

4. Construction of Semantic Keyword-based Search

The key idea behind the semantic keyword-based search contains two parts: 1) building up the stem set for the keyword set using a certain stemming algorithm; 2) designing an efficient semantic keyword-based search scheme based on the construction of the stem set, which can return all the documents containing the semantically close keywords related to a given query word.

4.1. Stem Set Construction

In our scheme, we choose the Porter Stemming Algorithm [22] as our stemming algorithm to ascertain the root of the word. Porter algorithm aims at removing the common morphological and inflexional endings from the word to find its root form. For example, for the following set of words: "walk", "walks", "walking", "walked", they all have a similar meaning and their stem is "walk". In this stemming algorithm, there is a set of predefined rules and the word needs to go through those rules before outputting the stem. Here there are two examples of rules:

if a word ends in "ies", replace "ies" with "i".

if a word ends in "sses", replace "sses" with "ss".

In summary, Porter algorithm is quite simple and widely spread. In our scheme, we use Porter algorithm to map all the semantically close words or different variants of a word to the same stem. For all the keywords W extracted from the plaintext document

collection D, we should conduct the stemming process to construct a stem set S, as shown in Figure 3. For the words with the same stem s_i, we denote $W(s_i)$.

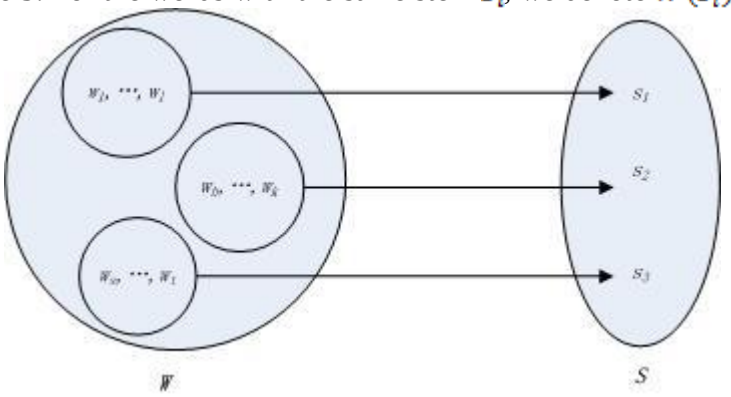

Figure 3. Stemming Process

4.2. The Efficient Semantic Keyword-based Search Scheme

Based on the stem set, we show the efficient semantic keyword-based search scheme in details. The scheme contains the following algorithms.

- **Setup** In this initialization phase, the data owner initiates the scheme to generate a random key $sk \xleftarrow{R} \{0,1\}^k$.
- **BuildIndex**
  - The data owner firstly computes $T_{s_i} = f(sk, s_i)$ for each $s_i \in S$, $1 \leq i \leq r$. Then he divides them into symbols as $T_{s_i} = \{\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,L/\theta}\}$ and builds up the symbol-based trie G covering all the stems of $s_i \in S$.
  - He attaches each $FID_{s_i} = Enc(sk, \{FID_{w_i} || w_i\}_{w_i \in W(s_i)})$ to G and outsources it, together with the encrypted document collection C, onto the cloud server.
- **GenQuery** For a given query word w_i, the data user performs the stemming process to attain the stem s_i. Then he computes $T_{s_i} = f(sk, s_i)$ and divides it into symbols as $T_{s_i} = \{\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,L/\theta}\}$. At last, he submits the search request T_{s_i} to the cloud server.
- **Search** Upon receiving the search request, the cloud server performs the search operation over the index G and returns FID_{s_i} to the data user. The user decrypts the returned results and retrieves relevant files.

5. Privacy Analysis

Document Privacy: The documents are encrypted separately before outsourcing and their confidentiality is guaranteed under the cipher. With secure enough cipher, it would assure that the encrypted documents leak no information except their length, size and document IDs.

Query Confidentiality: The trapdoor $T_{s_i} = \{\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,L/\theta}\}$ can be seen as a collection of L/θ symbols, which is unique through a hash function f. No information about s_i will be leaked from these symbols. Its confidentiality can be guaranteed by the one-way hash function f with the secret key sk, which is only shared among data owners and authorized users. Without knowing the secret key sk, the cloud server has no way to generate the valid trapdoor.

Index Privacy: Since each node in the trie G is the meaningless symbol $\alpha_i \in \Delta$, the cloud server cannot deduce the plaintext from the node. And each path from root to leaf node in G is L/θ long, which protects the length of real stem. There are both guaranteed by the one-way function f. For protecting the cloud server from knowing some statistic information from the shape of trie G, we can insert some dummy nodes to make the trie G to be a full $|\Delta|$-ary tree. And for protecting the cloud server from knowing frequency of stems from the length of FID_{s_i}, we can also insert some dummy items to make the length of each FID_{s_i} look the same.

6. Performance Analysis

In order to evaluate performance of our proposed scheme, we set up the experiment on a publicly available real dataset: the Enron Dataset [21]. We implement all the algorithms in our paper on a 2.83GHZ Intel Core(TM) processor, Windows 7 operating system with a RAM of 4G.

6.1. Stemming Process

In our experiment, we choose 10,000 documents randomly from the Enron Dataset as our document collection. In order to extract explicit keywords, we decide to make use of the method of "TF×IDF", which is the most widely statistical measurement in Information Retrieval. For each document, we compute the TF×IDF of each word, sort them in descending order and only choose top 5 words as the keywords of the document. After processing all the documents, we get a keyword set consisting of 20,419 keywords. In order to realize the semantic keyword-based search, we need to conduct stemming process on the keyword set and then get a stem set consisting of 18,711 distinct stems. Stemming process contains two operations: finding the stem of a word and grouping the words by stems. Figure 4 shows the time cost of stemming process is nearly linear with the number of keywords, and the complexity is $O(n)$. Note that conducting stemming process on 20,000 keywords just takes about 60s, which is quite efficient. Therefore, stemming process does not become a burden for the data owner and it is just a one-time cost before data outsourcing. Besides, we record the size of stem set in different size of keyword set in Table 1. Through stemming process, we can also reduce dimension of index by grouping the words with the same stem together.

Table 1. The Size of Stem Set in Different Size of Keyword Set

The size of keyword set	4000	8000	12000	16000	20000
The size of stem set	3861	7587	11272	14907	18535

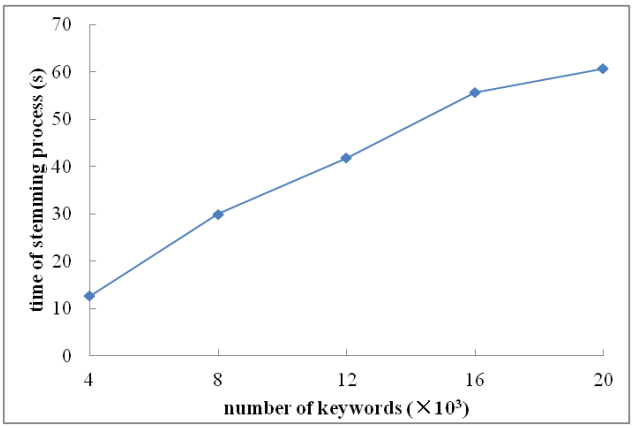

Figure 4. Time of Stemming Process

6.2. Index Construction

Index construction conducted in the data owner mainly involves: computing the hash value of each stem, dividing the hash value into symbols and building the symbol-based trie covering those symbols. In our experiment, we choose SHA-1 as the one-way function f and set θ as 4. Since the output length of SHA-1 is $L = 160$ bits, L/θ is 40 long which is height of symbol-based trie. Figure 5 shows that the time cost to build index is nearly linear with the number of stems, and thus the complexity is $O(r)$. Given the 18,711 stems, it just takes about 4.6s to build up the whole index. From the Figure 5, we can see index construction is quite efficient and just a one-time cost.

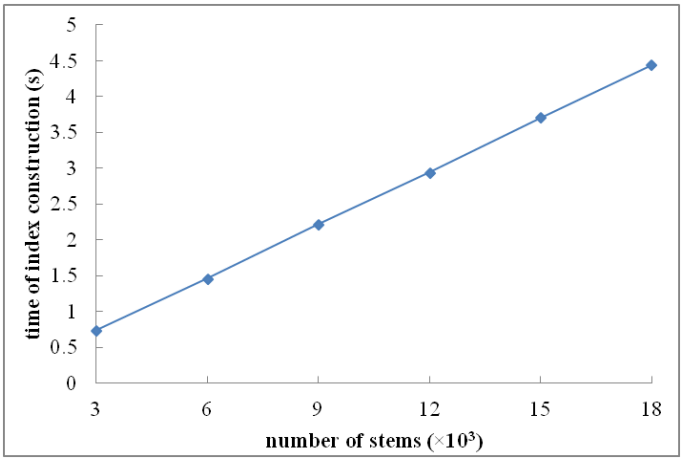

Figure 5. Time of Index Construction

6.3. Query Generation

Figure 6 shows that given different number of keywords in the query, the total time cost of query generation increases linearly. Query generation consists of stemming process on the query word, computing the hash value and division procedure. Given a single query word, it just takes about 2.5ms. And so query generation especially containing stemming process will not impose a burden on the data user.

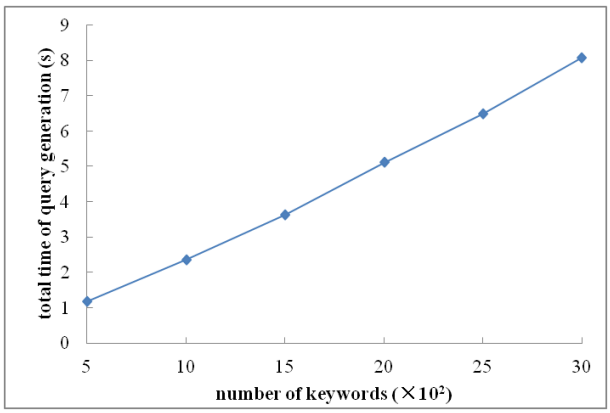

Figure 6. Time of Query Generation

6.4. Search

Given the symbol-based trie G covering 18,711 distinct stems, we measure the time cost of the search operation, as shown in Figure 7. In our scheme, since the index is based on the symbol-based trie, the time cost to search a trapdoor is decided upon the length of the trapdoor, rather than the size of document collection or the number of keywords in many existing searchable encryption schemes [10, 11]. That means, searching a trapdoor takes no more than 40 steps in our scheme. We get an estimation of throughput of search: 10,000 trapdoors/second. Note that searching irrelevant stems is much faster than searching relevant ones. The reason is that if there exists a mismatch between the symbols of the trapdoor and the index during the path exploration, the search operation will terminate at once. This "incomplete traversing" saves quite a lot of operating time.

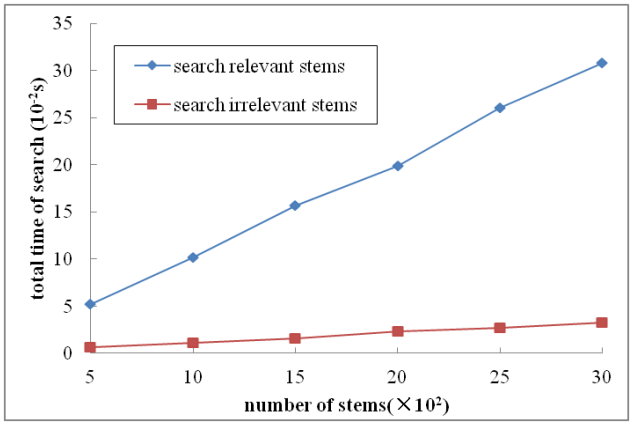

Figure 7. Time of Search

7. Conclusion

In the paper, we discuss and address the problem of querying different variants of a keyword. Combining with the stemming algorithm, we propose a semantic keyword-based search scheme over encrypted cloud data. Given a query word, data users can find all the documents containing the semantically close keywords or different variants through our scheme, which tackles the limitation of exact keyword search. Through rigorous privacy analysis and experimental study on real dataset, our scheme is quite secure and practical.

As our ongoing work, we will continue studying on semantic search over encrypted data from the following aspects: 1) designing a personalized search method based on the deep understanding of the user's search intention; 2) designing an intelligent search method for sentence.

Acknowledgements

This paper is a revised and expanded version of a paper entitled "An Efficiency Keyword Search Scheme to improve user experience for Encrypted Data in Cloud" presented at SoftTech 2014, Yeosu, Korea, May 8-10, 2014. This work is supported by the NSFC (61232016, 61173141, 61173142, 61173136, 61103215, 61373132, 61373133, 61402234), GYHY201206033, 201301030, 2013DFG12860, BC2013012, PAPD fund, 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), and China Meteorological Administration under CMA grants (2014 MC 16, M43).

References

[1] D. Song, D. Wagner and A. Perrig, "Practical techniques for searches on encrypted data", In Proceedings of S&P, **(2000)** May 14-17, Berkeley, CA.
[2] E.-J. Goh, "Secure indexes", Cryptology ePrint Archive, **(2003),** http://eprint.iacr.org/2003/216.
[3] Y. C. Chang and M. Mitzenmacher, "Privacy Preserving Keyword Searches on Remote Encrypted Data", In Proceedings of ACNS, (**2005**) June 7-10, New York, NY, USA.
[4] R. Curtmola, J. A. Garay, S. Kamara and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions", In Proceedings of the 13th ACM conference on Computer and communications security, **(2006)**.
[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky and G. Persiano, "Public key encryption with keyword search", In Proceedings of International Conference on Theory and Applications of Cryptographic Techniques, **(2004)** May 2-6, Interlaken, Switzerland.
[6] P. Golle, J. Staddon and B. R. Waters, "Secure conjunctive keyword search over encrypted data", In Proceedings of ACNS, **(2004)** June 8-11, Yellow Mountain, China.
[7] L. Ballard, S. Kamara and F. Monrose, "Achieving Efficient Conjunctive Keyword Searches over Encrypted Data", In Proceedings of International Conference on Information and Communications Security, (**2005**) December 10-13, Beijing, China.
[8] Y. H. Hwang and P. J. Lee, "Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System". In Proceedings of International Conference on Pairing-Based Cryptography, (**2007**) July 2-4, Tokyo, Japan.
[9] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data", In Proceedings of the 4th Theory of Cryptography Conference, **(2007)** February 21-24, Amsterdam, Netherlands.
[10] N. Cao, C. Wang, M. Li, K. Ren and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data", In Proceedings of IEEE INFOCOM, **(2011)** April 10-15, Shanghai, China.
[11] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren and W. J. Lou, "Fuzzy Keyword Search over Encrypted Data in Cloud Computing", In Proceedings of IEEE INFOCOM, (**2010**) March 14-19, San Diego, CA, USA.
[12] C. Liu, L. H. Zhu, L. Li and Y. Tan, "Fuzzy Keyword Search on Encrypted Cloud Storage Data with Small Index", In Proceedings of IEEE International Conference on Cloud Computing and Intelligence Systems, **(2011)** September 15-17, Beijing, China.
[13] C. Wang, N. Cao, J. Li, K. Ren and W. J. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data", In Proceedings of IEEE 30th International Conference on Distributed Computing Systems, (**2010**) June 21-25, Genoa, Italy.
[14] C. Wang, N. Cao, K. Ren and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data", IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 8, **(2012)** .
[15] M. Chuan and W. Hu, "Privacy-aware BedTree Based Solution for Fuzzy Multi-keyword Search over Encrypted Data", In Proceedings of 31st International Conference on Distributed Computing Systems Workshops, (**2011**) June 20-24, Minneapolis, MN,USA.

[16] C. Wang, K. Ren, S. C. Yu and K. M. R. Urs, "Achieving Usable and Privacy-assured Similarity Search over Outsourced Cloud Data", In Proceedings of IEEE INFOCOM, (**2012**) March 25-30, Orlando, Florida, USA.
[17] 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 Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, **(2013)**.
[18] Z. Shen, J. Shu and W. Xue, "Preferred keyword search over encrypted data in cloud computing", In Proceedings of 21st International Symposium on Quality of Service, (**2013**) June 3-4, Montreal, Canada.
[19] M Islam, M Kuzu and M Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation", In Proceedings of Network and Distributed System Security Symposium, (**2012**) February 5-8, San Diego, California, USA.
[20] C Liu, L Zhu, M Wang, and Y Tan, "Search Pattern Leakage in Searchable Encryption: Attacks and New Constructions", Cryptology ePrint Archive, **(2013),** http://eprint.iacr.org/.
[21] "Enron Email Dataset." https://www.cs.cmu.edu/~enron/.
[22] M. F. Porter, "An Algorithm for Suffix stripping", Automated Library and Information Systems, vol. 14, no. 3, (**1980**).

Authors

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

Xingming Sun, he 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 College 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.

Lu Zhou, she received her BE in Software Engineering from Nanjing University of Information Science and Technology, China, in 2012. She is currently pursuing her MS in computer science and technology at the School Of Computer and Software, Nanjing University of Information Science and Technology, China. Her research interests include network and information security, steganography, digital watermarking, copyright protection technology.