SCIENCE CHINA
Information Sciences

• LETTER •

March 2020, Vol. 63 139105:1–139105:3
https://doi.org/10.1007/s11432-018-9525-y

Two-stage index-based central keyword-ranked searches over encrypted cloud data

Hong ZHONG[1,2,3], Zhanfei LI[1,2,3], Yan XU[1,2,3], Zhili CHEN[1,2,3] & Jie CUI[1,2,3*]

[1]*School of Computer Science and Technology, Anhui University, Hefei* 230039, *China;*
[2]*Institute of Physical Science and Information Technology, Anhui University, Hefei* 230039, *China;*
[3]*Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei* 230039, *China*

Received 19 May 2018/Accepted 16 July 2018/Published online 10 February 2020

Citation Zhong H, Li Z F, Xu Y, et al. Two-stage index-based central keyword-ranked searches over encrypted cloud data. Sci China Inf Sci, 2020, 63(3): 139105, https://doi.org/10.1007/s11432-018-9525-y

Dear editor,

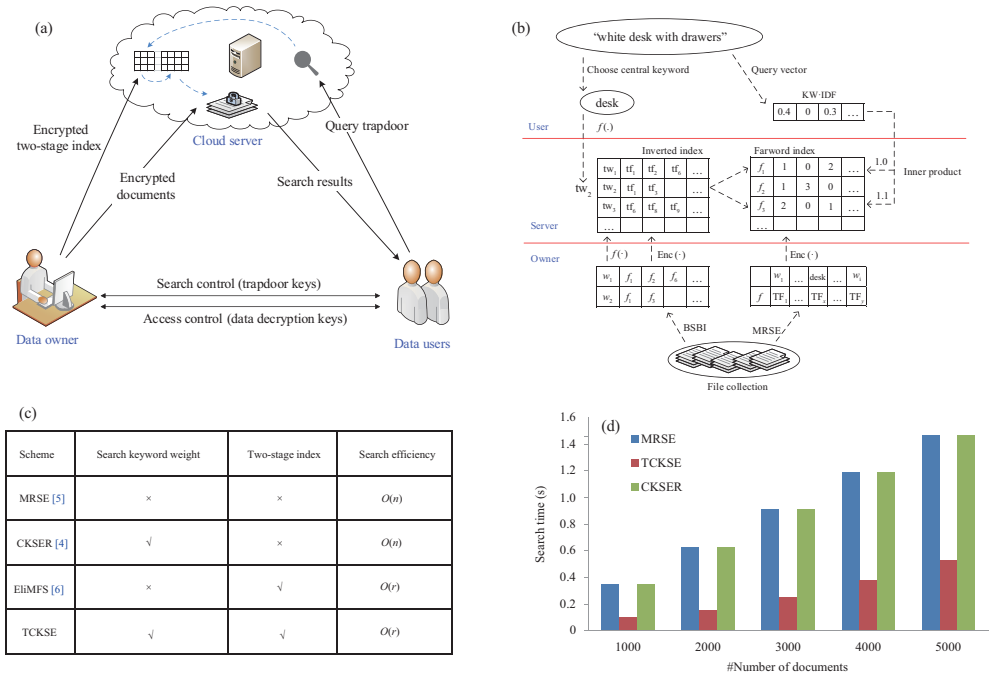
Searchable encryption [1] technology is an effective method for solving ciphertext keyword searches in cloud environments. In searchable encryption schemes, a variety of index structures are used to improve search efficiency. For example, Xia et al. [2] designed a dynamic searchable encryption scheme using a keyword-balanced binary tree, and the search efficiency was high. Wang et al. [3] proposed a multi-keyword search scheme based on an inverted index, which also improved search efficiency.

However, to provide search results more consistent with a user's search intentions, the importance of different query keywords must be distinguished. In 2017, Fu et al. [4] proposed the concept of the semantic expansion search of central keywords. Using a grammatical tree, they designed a keyword weight algorithm to calculate the weight of each query keyword, then selected the word with the maximum weight as the central keyword and utilized the synonyms extension. The keyword weight and IDF are inserted in the query vector, rendering the search result more consistent with the user's actual search intention. However, their scheme did not ensure that the search results contained the central keyword; as such, the search results may contain other keywords but no central keyword. Meanwhile, the search process needs to traverse all the index vectors and calculate the inner product, causing large overhead.

Thus, we propose a central-keyword ranked search scheme based on a two-stage index. Our scheme uses an inverted index combined with a forward index to solve the problems mentioned above. Compared with a single index method, our approach can greatly reduce the computation cost of deriving the vector inner product in the query process, thereby improving search efficiency. In addition, for multiple query keywords, we ensured that the search results contained the central keyword. Hence, the search results of our scheme are more accurate.

Model and algorithms. The system model and framework are shown in Figure 1(a) and (b). There are three participants: data owners, data users, and the cloud server. The data owners are responsible for several tasks; for example, the pre-processing file set $F = \{f_1, f_2, \ldots, f_n\}$, building inverted and forward indices, generating the encrypted two-stage index I, and uploading the encrypted two-stage index I and encrypted file set C to the cloud server. To search keywords, users must obtain the authorization of a data owner by obtaining the trapdoor generation key SK and the file decryption key. To search for encrypted files corresponding to t query keywords, users must construct query vectors, encrypt them using the trapdoor generation key SK, generate the query trapdoor T, and send them to cloud servers. The cloud server is responsible for storing the encrypted data of and providing the key-

* Corresponding author (email: cuijie@mail.ustc.edu.cn)



Scheme	Search keyword weight	Two-stage index	Search efficiency
MRSE [5]	×	×	$O(n)$
CKSER [4]	√	×	$O(n)$
EliMFS [6]	×	√	$O(r)$
TCKSE	√	√	$O(r)$

Figure 1 (Color online) (a) The system model of TCKSE; (b) the framework of TCKSE scheme; (c) comparison of related schemes; (d) search time VS the number of documents.

word search service for data users. When receiving the query trapdoor T from users, the cloud server searches on the encrypted two stage index I to derive the file identifier set id, then returns the encrypted target file to the user according to file identifiers. Finally, the user decrypts the query result. The details of the scheme are as follows.

(1) Setup. Input security parameter m, λ, data owner generates secret key $\mathrm{SK}(M_1, M_2, S, k_1, k_2)$, where M_1 and M_2 are $(m+2)\times(m+2)$ invertible matrix, S is an $(m+2)$-bit random vector, k_1, k_2 are intermediate keys, $k_1, k_2 \leftarrow \{0,1\}^\lambda$.

(2) Building the two-stage index. (a) Data owner extracts the keyword set $W = \{w_1, w_2, \ldots, w_m\}$ from the file set $F = \{f_1, f_2, \ldots, f_n\}$, and generates an inverted index. Next, the keyword tag $\mathrm{tw}_i \leftarrow f(w_i, k_1)$ and key $k_e \leftarrow f(w_i, k_2)$ are generated for each keyword w_i in the inverted index using the pseudorandom function $f(\cdot): \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$. Following this, file tags are generated for file identifiers corresponding to each keyword, and the encrypted inverted I_1 index is obtained. (b) A data owner generates an m-bit index vector v for each file f, if the keyword $w_x \in W$ is contained in the file f, insert TF of keyword w_x in the corresponding bit of vector v, namely $v_x = \mathrm{TF}_{w_x}$. Next, the vector is extended to $(m+2)$-bit, where the random number ε is inserted in the $(m+1)$th-bit and the number 1 is inserted in the $(m+2)$th-bit, namely $v' = (v, \varepsilon, 1)$. Finally encrypts the index vector v' by key SK. The index vector v' is divided into two vectors $v'1$, $v'2$ according to the random vector S. If $S_i = 1$, $v'1_i = v'2_i = v'_i$, and if $S_i = 0$, $v'1_i = \frac{1}{2}v'_i + r$, $v'2_i = \frac{1}{2}v'_i - r$, where r is a random number. Following this, the encrypted index vector $v'' = (M_1^{\mathrm{T}} \cdot v'1, M_2^{\mathrm{T}} \cdot v'2)$ is obtained by the matrix calculation. Finally, data owner gets the forward index $I_2 = \{v''_1, v''_2, \ldots, v''_n\}$. (c) The data owner uploads the two-stage index $I = \{I_1, I_2\}$ and the encrypted file set C to the cloud server.

(3) Generating the query trapdoor. (a) Data users calculate the keyword weight $\{\mathrm{KW}_1, \mathrm{KW}_2, \ldots\}$ of each query keyword using a keyword weight algorithm and select the keyword with greatest weight as the central keyword. The tag tw is generated for the central keyword w using the pseudorandom function $f(\cdot): \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$. (b) The m-bit query vector q is generated, where each bit represents a corresponding keyword in the keyword dictionary, and the initial value is 0. Next, the weight of each query keyword is inserted into the query vector q, namely $q_i = \mathrm{KW}_i \cdot \mathrm{IDF}_i$. Following this, the vector is extended to $(m+2)$-bit, thereby, providing $q' = (rq, r, t)$, where r, t are random number. Finally, the query vector if encrypted by splitting the query vector q' according to S. If $S_i = 1$, $q'1_i = \frac{1}{2}q'_i + r$, $q'2_i = \frac{1}{2}q'_i - r$; and if $S_i = 0$, $q'1_i = q'2_i = q'_i$. Then the encrypted query vector $Q = (M_1^{-1}q'1, M_2^{-1}q'2)$ is obtained by the matrix calculation. (c) The query trapdoor

$T = \{\mathrm{tw}, k_e, Q\}$ is uploaded to the cloud server by the users.

(4) Two-step search. (a) The cloud server searches the target file tag $\mathrm{tf} = \{\mathrm{tf}_x, \mathrm{tf}_y, \ldots\}$ in the inverted index I_1 using the central keyword tag tw. (b) The file tag is decrypted and the file identifier $\mathrm{id}_x \leftarrow \mathrm{Dec}(\mathrm{tf}_x, k_e)$ is retrieved. Next, the inner product between Q and the index vector is calculated to derive the relevance score, as follows:

$$\begin{aligned}\mathrm{Score} &= Q \cdot I_2\\ &= (M_1^{-1}q'1, M_2^{-1}q'2) \cdot (M_1^{\mathrm{T}} \cdot v'1, M_2^{\mathrm{T}} \cdot v'2)\\ &= q'1 \cdot v'1 + q'2 \cdot v'2\\ &= (rq, r, t) \cdot (v, \varepsilon, 1)\\ &= r(q \cdot v + \varepsilon) + t.\end{aligned}$$

Finally, the encrypted target file set is identified using the file identifier and the ranked result is returned to the user.

Scheme analysis. The comparison of related schemes is shown in Figure 1(c), where n is the total number of files and r is the number of files containing searched keywords. Our scheme (TCKSE) and Fu et al.'s scheme (CKSER) both consider the weight of search keywords and emphasize the importance of different search keywords, while MRSE and EliMFS [5] do not. Compared with these methods, our scheme is more in line with users' search intentions. Furthermore, our TCKSE method and Chen et al.'s EliMFS method both build the two-stage index structures, the other two schemes do not adopt this approach. EliMFS employs bloom filter and locality-sensitive hashing to generate index vectors and support fuzzy keyword searches. Our scheme and MRSE [6] use a keyword dictionary to generate index vectors, which leads to with higher accuracy, owing to false positives from the bloom filter. In addition, our scheme and EliMFS use an inverted index as the first stage index, meaning the search complexity is $O(r)$, which is a significant improvement upon the MRSE and CKSER schemes. Thus, our scheme is more accurate than EliMFS, and is more efficient than both CKSER and MRSE.

We selected the request for comments file set for testing. In our experiment, we used 5000 files and selected 5000 keywords from 5807 extracted keywords. The programming language is Java and the machine is Windows 7 system with an Intel Core i5 CPU (3.0 GHZ, 8 GB RAM). As shown in Figure 1(d), the search time of MRSE and CKSER are linearly related to the size of the file set. Because the cloud server must traverse all file index vectors to perform a search, the search complexity is $O(n)$. However, due to the two-stage index structure adopted in our scheme, after a first-stage index (inverted index) screening, only a few index vectors are traversed on the second-stage index to obtain search results; the search complexity is $O(r)$. Thus, search efficiency is significantly improved.

Conclusion. We investigated a central keyword search problem based on a two-stage index, which we constructed by employing inverted and forward indices. By assigning weights to the keywords in the query phrase, the central keywords were selected and subsequently searched on the two-stage index. Our proposed method ensured that the search results contained the central keyword. Moreover, owing to the screening of the first stage on the inverted index, the computation cost of the inner product at the second stage was reduced, thereby minimizing the overall search time.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61572001, 61872001, 61702005).

References

1 Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proceedings of Security and Privacy, 2000. 44–55
2 Xia Z H, Wang X H, Sun X M, et al. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans Parallel Distrib Syst, 2016, 27: 340–352
3 Wang B, Song W, Lou W J, et al. Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee. In: Proceedings of International Conference on Computer Communications, 2015. 2092–2100
4 Fu Z J, Wu X L, Wang Q, et al. Enabling central keyword-based semantic extension search over encrypted outsourced data. IEEE Trans Inform Forensic Secur, 2017, 12: 2986–2997
5 Chen J, He K, Deng L, et al. EliMFS: achieving efficient, leakage-resilient, and multi-keyword fuzzy search on encrypted cloud data. IEEE Trans Serv Comput, 2017. doi: 10.1109/TSC.2017.2765323
6 Cao N, Wang C, Li M, et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings of International Conference on Computer Communications, 2011. 829–837