Fault-tolerant Verifiable Keyword Symmetric Searchable Encryption in Hybrid Cloud

Jie Wang, Xiao Yu, and Ming Zhao
(Corresponding author: Ming Zhao)

Aviation University of Air Force, 130022, Changchun, Jilin, China
(Email: michaelwangliu@163.com & zhaoming2014@mail.jlu.edu.cn)
(Received Nov. 26, 2014; revised and accepted Feb. 10, & Mar. 16, 2015)

Abstract

As cloud computing is increasingly expanding its application scenario, it is vital for cloud storage customers not to sacrifice the confidentiality of sensitive data while making fullest use of operational functionality of cloud secure systems. Although traditional searchable encryption can well solve exact keyword search on encrypted data with retrieving files by search interest, it does not work when typos or misspelling mistakes occur. Many specific algorithms have been well proposed to solve this difficult problem. However, most of the schemes mainly focus on the single cloud to achieve fuzzy keyword search, which means that fuzzy-keyword index construction must take possible typos into account and makes existing exact-keyword index useless. In addition, existing searching schemes rarely take interaction between the data user and the cloud to improve system's usability and user's retrieval satisfactory degree into consideration. In this paper, we propose an improved scheme named as Distributed Fault-tolerant Keyword Search Supporting Verifiable Search-ability (DFKSSVS) in hybrid cloud with the emphasis of interaction circumstances. Through improved dictionary-based keyword construction scheme, we generate fuzzy keyword set, and build secure index for efficient fuzzy search. After searching procedures, the scheme can support verifiability of returned files via *proof* returned by cloud as well, and interaction between data user and private cloud to achieve dynamic ranking of retrieval results statistically. Through rigorous security and thorough analysis, we show that the improved solution can meet verifiable fuzzy keyword search on cloud encrypted data with supporting the exact-keyword index already generated. Security analysis and extensive experimental results demonstrate the accuracy and efficiency of our proposed scheme.

Keywords: Cloud storage, fault-tolerant keyword search, improved-dictionary-based fuzzy set, outsourcing data, searchable encryption, verifiable keyword search

1 Introduction

Nowadays, the increasing growth of Big Data in IT industry impels the application expansion of cloud computing. As a typical application, cloud storage has gained popularity in many corporations and companies around the world. However, as the large amount of sensitive data, such as enterprise basic files, government investigation reports, private health records and so on, is in the out-of-control domain, data privacy has become the top concern of whether it is a must to outsource data to the cloud. Data encryption is an effective solution to keep its confidentiality, which has yet sacrificed data usability. Encryption can preserve outsourced data's confidentiality, integrity and accessibility (CIA) of cloud data, and no one can know the contents of encrypted files without decryption keys, however, secure cloud system usability is lowered for non-operation-ability on cipher-text.

The best solution for encrypted data computation is Fully Homomorphic Encryption (HHE), which allows users to operate directly on cipher-texts and then produce results of matching procedures. Gentry et al. [8] made a breakthrough in theoretical domain, but the scheme of construction efficiency is far from practical utilization. Moreover, data users are usually interested in the most relevant files whose ranking is in the top-k list rather than all files returned from cloud. From the perspective of information retrieval, users choose to input some specific keywords named as "keyword-based search" to selectively retrieve relevant files. Unfortunately, computable retrieval operations on encrypted data by keyword search are limited due to no suitable schemes on cipher-text search compared with traditional retrieval methods on plain-text search. Although encrypted keyword can protect its privacy, how to use plain-text search techniques on encrypted data turns to be a real problem, which attracts much more attention of researches on it. Different from the traditional Private Information Retrieval (PIR) schemes, an alternative, that is searchable encryption (SE) schemes, has been proposed and researched for a long time. SE is a key technique for data users to di-

rectly operate encrypted data, but traditional SE without secure index is inefficient facing with the large-scale cloud encrypted data. So, it is important to construct a secure index for encrypted data files.

Searchable encryption is an important and fundamental solution to solve the problems of encrypted data utilization, as well as integration of data confidentiality and usability. In general, searchable encryption can be divided into two subcategories, that is, Public Key Searchable Encryption (PKSE), and Symmetric Searchable Encryption (SSE). PKSE can support much more flexible search operations and complicated search applications, but more computationally huge overhead is produced because of many pairing-operations compared with SSE. In contrast, SSE depends on its computational efficiency and operational convenience to make it to be a research hot spot. No matter PKSE or SSE, keyword search is associated with index of files. By integrating trapdoors (encrypted form of searched keyword) with secure index (encrypted form of file index), effective keyword search can be finished while retrieval contents and search results are blind to cloud servers.

Furthermore, fuzzy keyword seems to be a hot topic in the plain-text information retrieval field, because retrievers may have typos by accident or statistical misspelling mistakes during retrieval procedures. As an applicable expansion, fuzzy keyword search on encrypted data has been researched actively. Li et al. [11] for the first time proposed wildcard-based keyword search over encrypted scheme, which is proven its weakness of insecurity by Zheng et al. in HPCC 2013 conference [20]. Wang et al. [18] suggested a solution using trie-tree for index construction, which has large space-cost of building index and infeasible updating of index tree. Chuah et al. [6] presented a scheme which has secure index by using bed-tree with low efficiency. Liu et al. [12] proposed a solution named as "dictionary-based fuzzy keyword search on encrypted data" with small index, but its fuzzy keyword set is not all-around, which means loss of many possible exact keywords to match with. Recently, Zhou et al. [21] proposed a different scheme to make fuzzy keyword set by utilizing k-gram. Wang et al. [16] aimed at multi-keyword fuzzy search on encrypted data by locality sensitive hashes and Bloom filters to support multi-keyword search with low search complexity. However, all the schemes face with retrieval efficiency problem and defective construction of fuzzy-keyword index which has made exact-keyword index already constructed useless.

Another issue to which needs to pay much attention is verifiability of returned encrypted data from the public cloud. This was first mentioned by Chai et al. [5], who proposed a new searchable encryption scheme called VSSE. Due to the fact that it is unknown that the public cloud may save computation or download bandwidth for its selfishness, the returned encrypted data may be only a fraction of all retrieval outcome. So, verifiable searchability as well as protection of data confidentiality is a real applicable scenario during fuzzy search on cloud encrypted data. Wang et al. [19] has found the combination of fuzzy keyword search and verifiable keyword search on encrypted data and proposed a new scheme named VF-SSE, which means that data user can verify the correctness and completeness of returned files after the fuzzy search has already completed corresponding with a query containing a keyword of little typos. However, Buildindex phase in his scheme is conducted by data owner using wildcard-based scheme, which means that data owner may abandon the exact-keyword index constructed before and generate a specialized fuzzy-keyword index for fuzzy searching, thus it is inevitable for data owner to waste much more computation and storage resources. Another issue in his scheme is that ranking of keyword-retrieval has not been well tackled, and his work is mainly on the public cloud setting without applying in the hybrid cloud circumstances.

Based on thorough analysis on existing fuzzy keyword search schemes, we propose a novel scheme totally different from previous work. In this paper, we mainly concentrate on verifiable fault-tolerant keyword search on the cloud encrypted data and suggest a solution, which is called Distributed Fault-tolerant Keyword Search Supporting Verifiable Search-ability (DFKSSVS), to build secure exact-keyword index supporting verifiability in the public cloud, as well as generate fuzzy keyword trapdoors for matching in the private cloud. Due to Li's scheme weakness of insecurity, we abandon the scheme of directly using wildcard-based method to construct secure index, but we adopt traditional exact-keyword searching scheme. Our scheme will reduce index generation and storage complexity and guarantee highly efficient retrieval, and it can make fullest use of computation and storage resources in the private cloud. Our contributions of this paper can be summarized as follows:

1) We propose a novel **DFKSSVS** scheme in the hybrid cloud. We define the system and threat model, which means to be "semi-honest-but-curious" in the public cloud, and "honest-but-curious" in the private cloud. Preliminaries have been denoted to depict **DFKSSVS** scheme in detail.

2) In the public cloud, we use the exact-keyword index, which is already built for exact keyword search, or build exact-keyword index for searching for its first time. To reach verifiable search-ability, we use trie-tree based on symbol set, where a multi-way tree is constructed for storing a certain keyword trapdoor which can be recovered from the root node to the leaf node. Updating of index can be easily done on the tree structure according to trapdoor revising requests. Exact keyword search throughout the index can be well done, and encrypted data stored can be returned as well.

3) In the private cloud, we make use of the potential computation and storage resources to generate fuzzy keyword set, and trapdoors corresponding with ex-

act keywords responsible for searching on the exact-keyword index in the public cloud. At the same time, we allow statistically dynamic ranking of exact elements through feedback scheme in order to return more encrypted files related to data user's input keyword. Also, decryption of returned encrypted files is conducted and completed in the private cloud and it outputs plain-text files to data user.

The remainder of the paper is organized as follows: Section 2 introduces the system model, treat model, design goals, and preliminaries. Section 3 presents the novel scheme **DFKSSVS** in detail. Section 4 gives the security analysis of the whole scheme. Section 5 gives performance evaluation compared with [5, 11, 19] respectively. Related work for searchable encryption SE is discussed in Section 6. Finally, Section 7 concludes the paper.

2 Problem Statement

2.1 System Model

In the paper, we consider a cloud setting consisting of the entities: Data User (DU), Data Owner (DO), the Public & Private Cloud (PC), as is illustrated in Figure 1. Given a collection of n files denoted as F and their encrypted forms denoted as C, exact keyword set W extracted from F, secure index for C derived from W, the private cloud can generate fuzzy keyword set as well as trapdoors, which are produced with the secret key generated by authorized DUs, of similar keywords for exact matching in the public cloud, and the private cloud receives encrypted files corresponding with trapdoors, decrypts and returns the plain-text form of them to DU. Here, we denote that the private cloud can provide as much computation capability as possible just for relieving DU's burden of computing and storage. The public cloud, which is responsible for mapping trapdoors to encrypted files indexed by their IDs and linked to a series of exact keywords, supports exact matching throughout the secure index and returns encrypted files to the private cloud, and it has verifiable search-ability due to DU's verifying request to the series of searching procedures. DO has files needed to be outsourced to the cloud, and generates secret keys through $Setup(k)$ phase, which are shared with authorized DUs. DU raises a query and verifies the correctness and completeness by *proof* sent from the public cloud. In all, our **DFKSSVS** scheme makes the fullest use of specific characters of different parties in the hybrid cloud and certainly applies the real circumstance of keyword search over a large scale of cloud encrypted data.

2.2 Threat Model

Firstly, we assume that authentication between DO and DU has been appropriately done. To search relevant files for a certain keyword, the trapdoor, which is of encrypted form, of the given keyword must be generated so as to match items throughout the secure index in the public cloud. And DU may want to verify the completeness of retrieval results by sending requests to the public cloud. Here, we consider the private cloud to be "honest-but-curious", which means the private cloud servers honestly obey the principles of different protocols, and have the ability to learn something additionally sensitive information, at the same time, the public cloud to be "semi-honest-but-curious", which means the public cloud servers may be selfish in order to save computation and bandwidth of its own, and have the same basic characters of private cloud servers. In addition, we take Known Ciphertext Model into consideration, which means the cloud can only have access to encrypted files, secure index and trapdoors, without leaking any information but search pattern and access pattern. The semantic meaning of the model with its proven-security has been proposed in [7].

2.3 Design Goals

To enable normally searching on encrypted data when typos occur, we need to do some work in the trapdoor generation procedure in order to match corresponding items throughout the index tree which is already constructed before. Furthermore, the exact-keyword index can support verifiable-searching, so we choose the basic idea of trie-tree index based on symbol tree proposed in [18], and we utilize it in the exact-keyword searching circumstance. Specifically, we have the following goals:

1) Cipher-text search supporting fault-tolerant keyword-based query: this is the basic problem to which the paper is referred, fuzzy keyword search on secure exact-keyword index constructed before is always supported as well.

2) Verifiable-searching in the cloud: this is the need of DU who wants to verify the correctness and completeness of retrieval results of a given input keyword by the *proof* returned from the cloud.

3) Keyword privacy: in spite of leak of search pattern, the cloud should not deduce any sensitive information through secure index, encrypted trapdoor, and encrypted files, as is requested to be securely encrypted to minimize information leak risks.

4) Privacy guarantee: encrypted files should be returned to DU if and only if the correct trapdoor of a given keyword generated by authorized DU with the secret key matches the items in secure index and file IDs are obtained to link with the encrypted files.

5) Result accuracy: By feedback scheme can exact keywords in the fuzzy keyword set achieve ranking dynamically in the private cloud, which is helpful with trapdoor matching with its ranking position forward on the score list. This is similar with statistical methods, but it will not leak no more information than search pattern and access pattern by encryption.

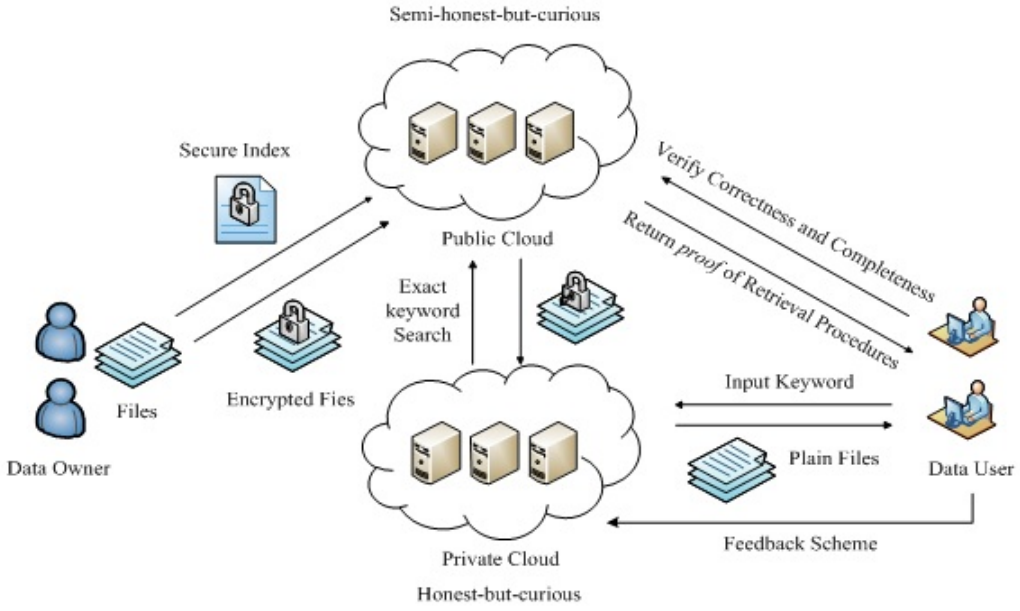

Figure 1: Architecture of system & threat model

2.4 Preliminaries

Edit distance: Given two strings S_i and S_j, the edit distance between them is defined as $ED(S_i, S_j)$, which means the minimum steps from one string to another, including insertion, deletion, and substitution of some character in the string [13].

Trie-tree: A trie-tree, which is always described as prefix-tree, is a data structure with its essential order of contents in each node storing associative array where keys are always strings. Different from a binary search tree, the node of trie-tree in each position presents the key of an array, sharing the same prefix stored in its parent's node. And the root node is always associated with an empty string regarded as the starting point to conduct searching.

Hash function: A hash function is a function used to map any arbitrary size of data to a fixed size, with slightly different input giving rise to big difference of its output. Here in this paper, we use collision-resistant hash function (MD5, SHA-1) to generate trapdoors of exact keywords in the fuzzy keyword set, and to be the main function in the preprocessing procedure before constructing secure index outsourced to the public cloud.

Dictionary: it is a pre-defined keyword collection which is consisted of all indexed items (keywords) linked with certain encrypted files.

Typical algorithms: our scheme **DFKSSVS** is composed of six polynomial-time algorithms denoted as $KeyGen(1^k)$, $BuildIndex(sk,\ W_i)$, $ExactTrapGen(sk,\ W_i)$, $Test(I,\ Trapdoor)$, $Verify(I,\ proof)$, $Feedback(sk,\ W_s^{"})$. More details of algorithms are described below.

3 Distributed Fault-tolerant Keyword Search Scheme Supporting Verifiable Search-ability

In this part, we describe more details on **DFKSSVS** scheme. Based on Wang's [19] and Chai's [5] schemes proposed before, we present our novel scheme getting greater effects supported by experiential analysis and experiments on real-world data set.

3.1 $KeyGen(1^k)$

In the process, DO generates secret keys for index and trapdoor generation, as well as the key for keyed hash function, and file encryption. The $KeyGen$ phase is a randomized key generation algorithm, which is set up and outputs keys in the way: $hk, tk, fk \xleftarrow{R} \{0,1\}*$, that is to say, we take k as input and different secret keys are output.

3.2 $BuildIndex(sk, W_i)$

In the process, we consider to use the symbol-based trie-tree to construct secure index I for the whole encrypted files of DOs. We use the scheme proposed by Chai [5] to achieve verifiable-searching on the index tree and integrate Li's [11] fuzzy keyword generation method to generate symbol-based trie-tree index based on exact keywords without abandon of exact-keyword index tree already constructed before. That is to say, we can use the new scheme

to complete fuzzy searching over cloud encrypted data on the generally secure index tree without verifiability of retrieval results, or on the trie-tree index supporting verifiability of matching procedure, and we can also achieve exact keyword searching on encrypted data as well.

- Keyword extraction from files.
  In this phase, DO extracts distinct keywords from plain-text files, which are used for constructing index for each file. Different keywords have their own keyword-weight in each file and one can distinguish a specific file from others by keywords that the file possesses. Here, we denote W as exact keywords extracted from the file set. We can also utilize the comprehensive dictionary D to check each element's correctness and completeness of the exact-keyword set.

- Build trapdoors of exact keywords.
  To exact-keyword set, we need to generate trapdoors of elements of the keyword set by a pseudo-random one-way function, which is always used keyed hash function $f_{hk}(\cdot)$. DO computes $T_w = \{T_{w_i}\} = \{f(hk, w_i), w_i \in W\}$ for each $w_i \in W$ with the index generation key hk. And then, DO divides each T_{w_i} into a series of n-length bits determined by its corresponding symbol in Δ, which is denoted as $\eta_1, \eta_2, \cdots, \eta_{z/n}$, where z is the output length of keyed hash function.

- Initialization of trie-tree index.
  Firstly, DO takes a quick scan of an empty trie-tree and determines that the root node is associated with an empty set as the beginning of searching throughout the whole tree. In addition, DO defines the identifier for every document file which is to be outsourced and obtains the identifier set $ID = \{ID_p, P = |F|\}$, as well as ID_{w_i} representing all file IDs containing w_i, which is a vital path to search for the relevant encrypted files corresponding with w_i.

- Build symbol-based trie-tree index.
  In this phase, we mainly focus on how to insert $\eta_1, \eta_2, \cdots, \eta_{z/n}$ into the trie-tree to achieve index construction of exact keywords in the set, as well as verifiability of retrieval results. To the root node, we define an empty set to be regarded as the beginning of keyword-searching. In every child node, we insert a two-tuple unit, one is η_i and the other is δ_i, which symbolizes the route from its parent node to its own and from itself to its child node. The content of δ_i is presented as $p_i||q_1||g_{ik}(p_i||q)$, where p_i is a bit-stream, which represents information of its parent node, of 2^m in which η_i corresponding with the position in Δ is set to 1 while other positions are set to 0, q_i represents information of its child node with the same way as mentioned above. g_{ik} is a keyed hash function to encrypt node information to support verifiable-searching. For example, if the current node stores η_a, whose parent node stores η_b with its position being the b-th symbol in Δ, and child node stores η_c in the same way. Then, $p_i = 0, 1, 0, 0, \cdots, 0$, $q_i = 0, 0, 1, 0, \cdots, 0$. More detailed information in all is depicted in Algorithm 1.

Algorithm 1: $BuildIndex(sk, W_i)$

Require:
(1) secret keys $hk, tk, fk \xleftarrow{R} \{0,1\}^*$
(2) exact keyword set containing keywords extracted from files by
(3) files needed to be outsourced Dictionary D
Ensure:
Trie-tree index
1. create an empty tree for secure index
2. initialize (null, null) for each node
3. **for** each $w_i \in W$ **do**
4. compute $T_W = \{T_{w_i}\} = \{f(hk, w_i)\}_{w_i \in W}$
5. cut each T_{w_i} into $\eta_1, \eta_2, \eta_3 \ldots \eta_{z/n}$
6. **for** each node in the trie-tree **do**
7. insert $\eta_j (j \in \{1, 2, 3 \ldots z/n\})$ to its node in $j-th$ layer
8. **if** the node is not leaf-node **then**
9. insert $\delta_j = p_i \| q_i \| g_{tk}(p_i \| q_i)$ at the same time
10. **else** insert $p_i \| ID_{w_i} \| g_{tk}(p_i \| ID_{w_i}) \| g_{fk}(ID_{w_i})$ to its leaf-node
11. **end if**
12. **end for**
13. **end for**
14. output the secure trie-tree index
15. encrypt F with fk and output C which is to be outsourced

- Files to be encrypted and outsourced.
  In this phase, files in the collection need to be encrypted by secret key fk and outsourced to the public cloud. The connection between encrypted files and their IDs should be well done in the cloud so as to retrieve relevant encrypted files back to DU by searching ID_{w_i} stored in the leaf node on the trie-tree index.

3.3 $ExactTrapGen(sk, W_i)$

In this phase, we discuss the issue of trapdoor generation corresponding with the input keyword when typos and minor mistakes occur at the beginning of the query. Considering that Li's fuzzy keyword set construction scheme has been proven weakness of its insecurity by Zheng et al. [20] in HPCC 2013 conference, which is due to mutual dependency of retrieval history, our scheme uses improved wildcard & dictionary-based construction scheme to generate trapdoors of keywords to avoid high history dependency of trapdoor relevance of distinct keywords. Because the whole search scheme is constructed in the hybrid-cloud, we assume that much work referring to some sensitive information can be done in the private cloud so as to make the fullest use of its scalable computing and storage resources even if it seems to be "honest-but-curious" for DU. We consider that plain-text keywords and secret keys are deleted after construction of exact keyword set as well as generation of exact keyword trapdoors, which means that

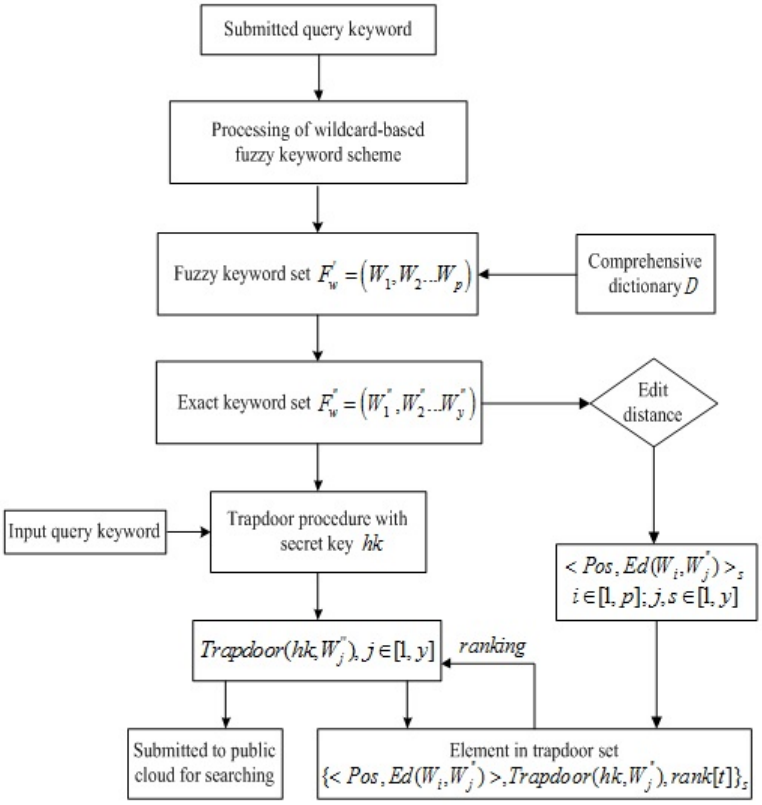

Figure 2: Exact-keyword trapdoor construction scheme

only encrypted forms of query information can be seen in the private cloud. Figure 2 is the exact-keyword trapdoor construction scheme.

In this process, we first build the fuzzy keyword set according to the submitted query by using Li's wildcard-based fuzzy keyword construction scheme to minimize the scale of its set, then through dictionary-based method, we can determine appropriate fuzzy keywords by substituting the wildcard with fixed alphabet, so we change fuzzy search into exact search so that keyword search on encrypted data can be completed in exact-keyword search scheme. Through comparison of edit distance, edit distance information with wildcard position Pos as its form of $< Pos, Ed(W_i, W_j') >_s$ can be well calculated to be an important part of ranking tuples which will be described below. Exact keywords in the fuzzy set will be transformed into trapdoors with secret key hk, which are submitted to the public cloud to conduct exact matching on secure index. We use improved dictionary-based fuzzy keyword construction scheme to expand the number of keywords in the set so as to absorb much more likely exact keywords for search, which can improve the probability of relevant encrypted files needed by DU. In addition, the private cloud sends $\{Trapdoor(hk, W_j')[\eta_1', \eta_2', \cdots, \eta_{z/n}']\}_{j \in [1,y]}$ which is generated in the public cloud back to DU so as to conduct $Verify(I, proof)$ procedure.

3.4 $Test(I, Trapdoor)$

Upon receiving search request from the private cloud, the public cloud server divides each trapdoor into a series of symbols in the same way mentioned in Algorithm 1. Then, Algorithm 2 can generate verification $proof$ containing ID_{w_i} back to DU, and returns relevant encrypted files to the private cloud. According to ID_{w_i}, DU can check the plain-text files output by the private cloud and make requests of verification of retrieval results to the public cloud. Detailed information is shown in Algorithm 2.

3.5 $Verify(I, proof)$

In this part, we introduce the verification process of retrieval results in detail. We have noticed that the secret key tk plays a very important role in the construction phase of δ_j. Given that each node in the trie-tree has a unique route from the root node to itself, we believe that we can verify correctness and completeness of results through the δ_j, which consists its unique parent node information and each child node symbol information. DO generates secret keys shared with authorized DUs, which makes attackers unable to forge a search $proof$ without the correct tk. And DU can verify retrieval results by re-generating $proof$ with shared secret key tk.

When the public cloud server completes the search pro-

Algorithm 2: $Test(I, Trapdoor)$

Require:
(1) secure trie-tree index
(2) trapdoors of exact keywords
Ensure:
Relevant encrypted files with their *ID*s and search *proof*
1. Cloud server divides $Trapdoor(hk, W_j^*)$ into a series of $\eta_1^*, \eta_2^*, \eta_3^* \ldots \eta_{sk}^*$ by Algorithm 1.
2. initialize *proof* to be $\varnothing$
3. **for** each $W_j^*, j \in [1, y]$ **do**
4. **for** each η_i^*, compare η_i^* with η_i **do**
5. **if** they are the same **then**
6. append δ_i to *proof*
7. conduct the next η_{i+1}^*
8. **else** append p_i to *proof*
9. break;
10. **end if**
11. **if** current node is leaf node **then**
12. append $p_i \| ID_{w_i} \| g_{tk}(p_i \| ID_{w_i}) \| g_{tk}(ID_{w_i})$ to its *proof*
13. set $proof \leftarrow proof \| j$
14. set $IDset \leftarrow ID_{T_w} \| j$
15. **end if**
16. **end for**
17. **end for**
18. output relevant encrypt files $C_{W_j^*}$ to the private cloud and *proof* to DU

cess, $IDset$ corresponding with relevant encrypted files, which are the file set of returned encrypted files from the public cloud, can be obtained by DU from the private cloud to check the integrity of retrieval results, and search *proof* is sent by the public cloud server. If the search process completes, DU can verify the results by $IDset$; otherwise, verification can be well done by *proof* contents wherever the search process is suspended. Another point needed to be noted is that the comprehensive symbol set Δ is shared with authorized DUs. See Algorithm 3 for more detailed verifying process.

3.6 $Feedback(sk, W_s'')$

In this part, we mainly focus on feedback scheme to construct a dynamic ranking list of trapdoors without leaking of sensitive information other than search pattern and access pattern in the private cloud. Considering that the private cloud always refers to cloud service for a specific organization or government, we believe that it is less "honest-but-curious" than what we assume to be in the common sense. So we can use some plain-text information corresponding with trapdoors of exact keywords to achieve dynamic ranking of retrievals statistically from DUs. By feedback scheme can DUs receive much more relevant files containing the exact keywords derived from the input keyword with minor typos, which is benefited from the effectively statistical tendency of typos or little mistakes between DUs and the private cloud. Figure 3 is a procedure of feedback scheme in **DFKSSVS**.

We have taken it into consideration that we should not

Algorithm 3: $Verify(I, proof)$

Require:
(1) $IDset^*$ from the private cloud
(2) *proof* from the public cloud
(3) secret key *tk* and shared symbol set Δ
Ensure:
Output "Right" for searching correctly and completely or "Wrong" otherwise
1. check $\{g_{sk}(\hat{IDset}_{w_i}^*)\}_{i \in [1,y]}$ with $\{g_{sk}(IDset_{w_i})\}_{i \in [1,y]}$
2. **if** they are equal **do**
3. output "Yes"
4. **else** "No"
5. **end if**
6. **for** each trapdoor $j \rightarrow 1$ to y returned from the private cloud **do**
7. **if** $t = z/n$ **then**
8. decrypt $g_{sk}(p_i \| ID_{w_i})$ to get p_i and its corresponding part in the trapdoor η_i^* of $Trapdoor(hk, W_j^*)$
9. **if** they are equal in terms with the position set to 1 in Δ **then**
10. output "Yes"
11. **else** "No"
12. **end if**
13. **end if**
14. **while** $t \geq 0$ **do**
15. decrypt $\{\delta_j = p_i \| q_i \| g_{sk}(p_i \| q_i)\}_i$ to get p_i and q_i
16. **if** $p_i = \eta_i$ and $q_i = \eta_{i+1}$ in terms with the position set to 1 in Δ **then**
17. output "Yes"
18. **else** "No"
19. **end if**
20. $j--$
21. **end while**
22. **end for**
23. **if** all the procedures output "Yes" **then**
24. output "Right"
25. **else**
26. output "Wrong"
27. **end if**

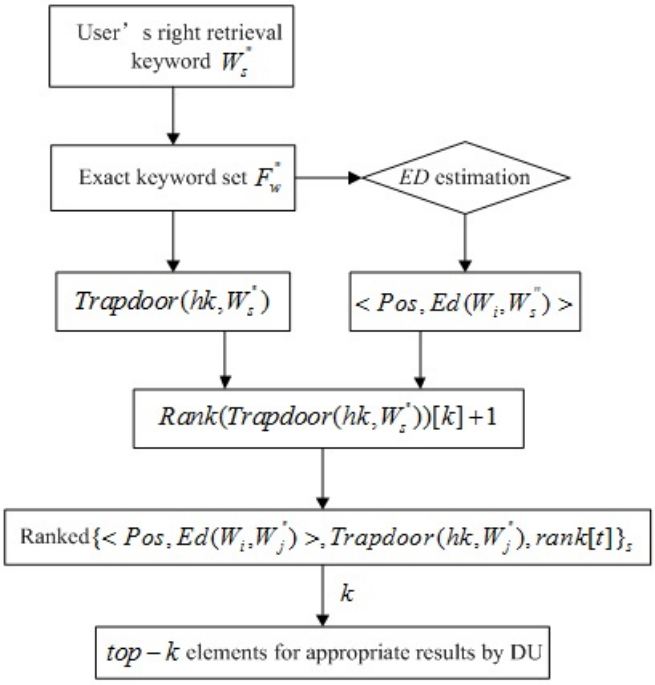

Figure 3: Procedure of feedback scheme in **DFKSSVS**

leak any sensitive information other than search pattern and access pattern, let alone the plain-text keywords existing in the private cloud. But we also emphasize the cloud being powerfully capable of computing and storage, so we choose exact keyword set with its trapdoors generation to be under control and they are to be deleted after the ranking element with its three tuples has been generated in the set. Above all, the search procedure is conducted by its encrypted form in the whole phase. Without loss of generality, we can find that the security of the novel search scheme on encrypted data has the same level as traditional SSE schemes researched before and achieves dynamically ranking of retrieval results between DUs and the cloud.

4 Security Analysis

Privacy-preservation: In this paper, we only take privacy-preserving concerns into account during the whole search procedures. For sensitive files, traditional encrypted algorithms can guarantee their security and integrity, which is out of discussion of our research. We mainly focus on confidentiality of the index and trapdoors in phases of $BuildIndex(sk, W_i)$, $ExactTrapGen(sk, W_i)$, and $Feedback(sk, W_s^{"})$. Due to security of collision-resistant keyed hash functions, generation of trapdoors can be securely conducted, which means that it is impossible for any attacker to get plain-text sensitive information without secret key hk and hash algorithms. In addition, in $Feedback(sk, W_s^{"})$ phase, although we expose plain-text keywords to the private cloud so as to exploit its potentially tremendous computing capability, we can assure that risks of privacy security can be reduced to its minimum point due to timely deletion operations of exact keywords in the private cloud, and lower security threat level compared with the public cloud as well. To gain dynamical ranking of retrieval results from DUs, we make the fullest use of the private cloud to compute exact keywords corresponding with the input fuzzy keyword and trapdoors of them with a little bit sacrifices of keyword privacy in the private cloud. Here, we refer to Li's scheme security analysis to prove our scheme's search security.

Theorem: the novel scheme is secure regarding to its search privacy.

Similarly with Li's method, we assume that the proposed scheme cannot achieve the index & trapdoor privacy against in-distinguish-ability under chosen-plaintext-attack (IND-CPA), which means that there exists a polynomial-time algorithm A who can intelligently deduce and rightly obtain plaintext sensitive information through the encrypted forms of keywords. Then, we construct another algorithm A' which utilizes A to decide whether $f'(\cdot) < g'(\cdot) >$ is a pseudo-random function the same as $f(sk, \cdot) < g(tk, \cdot) >$ or a real-random function. A' can have access to an oracle $O_{f'(\cdot)} < O_{g'(\cdot)} >$, and takes as a real number value x as input and $f'(x) < g'(x) >$ as the output. For any request of index & trapdoor generation, A' can answer it with $f'(\cdot) < g'(\cdot) >$. The

Table 1: Comparison of the aforementioned schemes

Content	Li's Scheme	Chai's Scheme	Wang's Scheme	Our scheme
Storage cost	$O(MN)$	$O(N)$	$O(MN)$	$< O(MN)$
Search cost	$O(1)$	$O(L)$	$O(1)$	$O(1)$
Construction cost	$O(MN)$	$O(N)$	$O(MN)$	$O(1)$
Verifiable-searching	NO	YES	YES	YES
Fuzzy-searching	YES	NO	YES	YES
Verification cost	-	$O(L)$	$O(1)$	$O(1)$
Supporting ranking	-	-	-	YES
Search intelligence	-	-	-	YES

adversary makes a request of two challenge keywords w_0 and w_1 after several queries to $O_{f'(\cdot)} < O_{g'(\cdot)} >$. A' takes a random $b \in \{0,1\}$ and submits w_b to the challenger for computing $f'(w_b) < g'(w_b) >$. Once A' receives the answer y, it sends y to the adversary to conduct a guess of answering the value of $b' \in \{0,1\}$. If A can get the right value of b' which is equal to b, $f'(w_b) < g'(w_b) >$ is not a random value, which can be directly deduced or easily guessed. By this way can A' decide whether $f'(\cdot) < g'(\cdot) >$ is a real random function or not. However, due to the standard assumption of in-distinguish-ability of pseudo-random functions and real-random functions, A can correctly guess the right value of b' with probability of 1/2, which makes it clear that the previous assumption is wrong. So, any keyword with its encrypted form in the search process can impossibly leak any sensitive information to the attack, that is, the search procedure is secure. Another point to which we should pay attention is that our provable-security process is a little bit different from Li's and Wang's methods whose variables are fuzzy keywords generated by wildcard-based scheme expansion. And $g(tk, \cdot)$ makes it almost impossible for any adversary to fake $\delta_j = p_i||q_i||g_{tk}(p_i||q_i)$ in each node, as well as $p_i||ID_{w_i}||g_{tk}(p_i||ID_{w_i})||g_{tk}(ID_{w_i})$ in the leaf node in the whole $proof$.

Verifiable-searching: DU can verify correctness and completeness of retrieval results by $proof$ which the public cloud sends back. The $proof$ is composed of several parts corresponding with the content of each node in the searchable trie-tree index. We decrypt each part in the $proof$ and compare its content with trapdoor's symbol element in the fixed position of Δ in order to check consistency of the both parts. Without tk, it is impossible for DU to conduct the procedure of verification of results, which means verifiable-searching can be securely achieved by DU.

5 Experimental Performance Evaluation

In this part, we first compare our novel scheme with Li's [11], Chai's [5], and Wang's [19] schemes so as to clearly present advantages of the scheme proposed in this paper. Here, we denote N as the total number of distinct keywords and M as the maximum size of exact keyword set generated through the input fuzzy keyword. Table 1 shows the comparative contents of the four aforementioned schemes. Secondly, we give the performance evaluations and analyses under the real-world set experiment.

5.1 Performance Comparison

In the aspect of storage cost, our scheme is different from Li's and Wang's schemes, which need $O(MN)$ cost when fuzzy keyword set was constructed. Given that the improved dictionary-based fuzzy keyword set construction method is used for exact keyword expansion, distinct keywords in the fuzzy set can be selected and filtered by the comprehensive dictionary so that the storage cost is cut down to less than $O(MN)$. In addition, although our scheme's search cost is the same as Wang's to be $O(1)$, it is more effective for locating the symbol in the set because we adopt to use symbol's position in the set to represent its content when secure index is generated. Due to our scheme's transportability, it is always feasible for secure exact-keyword-index constructed before to conduct exact matching for relevant encrypted files, so the construction cost can be a constant number $O(1)$ compared with the three schemes which need more computation and storage cost to generate the new index. With regard to verification cost, Chai's scheme require L times decryption operations to accomplish verification procedure, where L is the length of input keyword for fuzzy searching, but our scheme inherits Wang's scheme to achieve $O(1)$ verification cost by calculating hash value to match with the $proof$ sent from the public cloud. Furthermore, our scheme achieves interaction between the DU and the private cloud so as to make the returned encrypted files in order accordingly to statistical circumstances of mi-

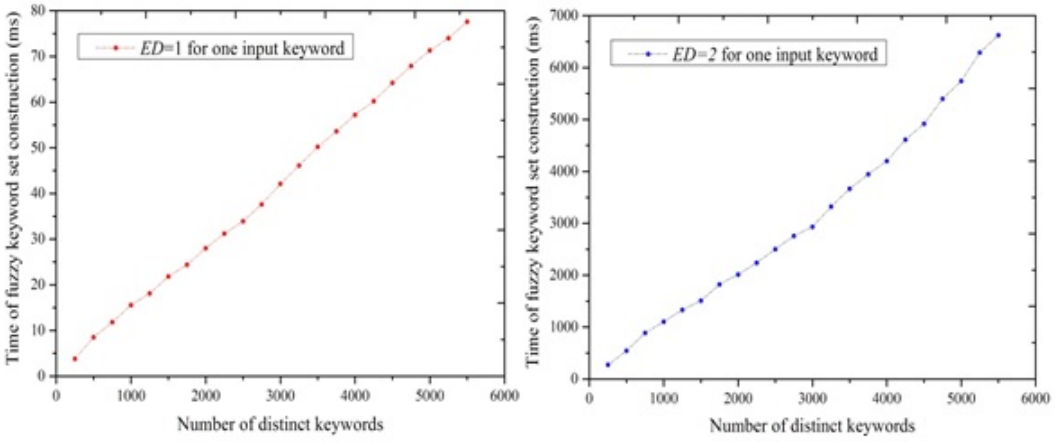

Figure 4: Time of fuzzy keyword set construction for $ED = 1$ (left) and ED=2 (right)

nor typos. This point can also achieve search intelligence through dynamic ranking of trapdoors in the private cloud in order to satisfy DU's needs of ranked retrieval results in the public cloud as well.

5.2 Performance Evaluation

In this section, we analyze the efficiency and accuracy of our scheme based on the experimental results in the real-world data set, that is, Request For Comments (RFC) which contains a large distinct keywords with technical files. And all the experimental results are obtained from implementation of the aforementioned schemes using JAVA on the Linux Server with Intel Core i5 Processor 4.0 GHz and 4G memory. Experimental contents include Time of Fuzzy Keyword Set Construction, Time of Index Construction, Time of Trapdoor Construction, Time of Keyword Search, and Recall & Precision of Top-k Results Corresponding with an Input Keyword, which can be also called as Performance evaluation.

Figure 4 presents time of fuzzy keyword set construction along with the number of distinct keywords for $ED = 1$ and $ED = 2$. We can easily understand that exact keywords generated by the method illustrated in Figure 2 are almost linear with the increasing number of distinct keywords for different edit distances. In addition, due to the fact that exact keywords in the fuzzy set are greatly expanded when $ED = 2$, the time of generating the fuzzy keyword set has reached 5.675s when the number of distinct keywords reaches 5000 in our experiment, which shows that edit distance is a key factor of shaping the overall efficiency of keyword search and one can no longer tolerate to waste a few seconds to generate exact keywords for fuzzy searching.

Figure 5 shows the relationship between time of index construction and number of distinct files. Because we adopt exact keywords extracted from the plain-text files to construct index, the time cost is mainly on calculating hash values of different keywords, positioning the corresponding symbols to insert inner nodes, and integrating each node's parent and child node position information to form δ_j for verification. We select [1000, 20000] files to extract and stem keywords for generating exact-keyword index, and the construction time is linear with the increase of distinct keywords corresponding with their files. Furthermore, our scheme always works well with regard to the secure exact-keyword index constructed before for searching relevant encrypted files in the public cloud.

Figure 6 gives the detailed information of trapdoor construction time along with the increase of distinct keywords. Similarly with Figure 4, the cost of trapdoor construction time is extremely large when $ED = 2$ because of the large number of exact keywords generated in the private cloud. We respectively select [1000, 10000] distinct keywords in RFC. Although edit distance is the main factor of keyword trapdoor cost, we can also find that $ED = 1$ is statistically much more common toward only one keyword. So we can take the instance of $ED = 1$ to be the main point into consideration without concern about low efficiency resulted from larger edit distances.

Figure 7 presents the time cost of keyword search. Given the real number of exact keywords indexed in the public cloud, here we choose [1000, 10000] respectively, we find that we convert fuzzy keyword searching into exact keyword matching so that the search time in terms of one input keyword has the same character as that of several times of exact matching of keywords. For all the procedures of keyword search, efficiency of our scheme can be accepted considering the existing schemes [5, 11, 16]. Moreover, because returned files include many relevant ones indexed by distinct keywords tracing back to the input keyword, the time can reach several seconds, which means that there are more selective ones for retrieving according to DU's searching interest.

Figure 8 shows the result of the proposed scheme supporting dynamic ranking, which embodies search intelligence by feedback scheme in the private cloud. Here, we use recall rate and precision rate to evaluate the whole

scheme. Recall rate is denoted as $t_p/(t_p+f_n)$ while precision rate is $t_p/(t_p+f_p)$, where t_p is true positive, f_p is false positive, and f_n is false negative. The value of k in $top-k$ selection is determined by DU according to his retrieval interest. In this experiment, we first set up the times of feedback to be 10, 50, 100, and choose the value of k in [10, 55] with step-length of 5 for $top-k$ selection. As illustrated in Figure 8, the recall rate is almost flat with the lowest percentage to be 96.42% no matter which k is selected, and the precision rate is markedly improved from $T = 10$ to $T = 100$ with its upper bound to be 98.57% in our experiment. Although different DUs have different retrieval interests, the interaction bridge between DU and the cloud can take DU's retrieval history with statistical typos into construction of dynamic ranking list for trapdoors of exact keywords without exposing any sensitive information other than search pattern and access pattern in the private cloud, which is a sparkling point of searching intelligence in symmetric searchable encryption (SSE) field.

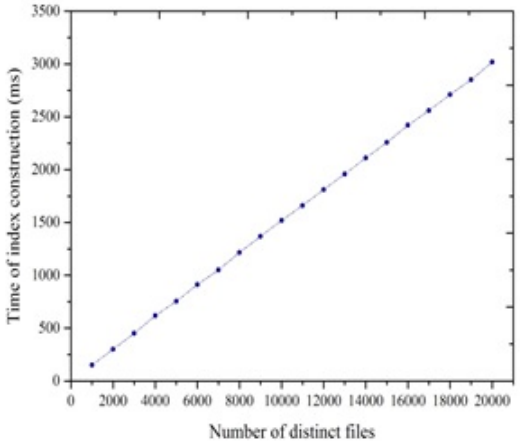

Figure 5: Time of index construction

.

6 Related Work

Song et al. [14] firstly presented the notion of searchable encryption with his specific stream cipher-text scheme. But the searching overhead is linear to the size of plaintexts which need to be encrypted. Goh [9] developed a scheme using Bloom Filter to minimize the work load under the condition of the number of all files in the collection set to establish a secure index. Boneh et al. [2] first constructed public-key based searchable encryption, whose work is so meaningful that many other scientific research teams propose different schemes achieving public-key encryption and private-key decryption. Conjunctive keyword search schemes are also recommended in [1, 3, 4, 10, 15], and specific real needs such as order-preserving symmetric encryption, single keyword or

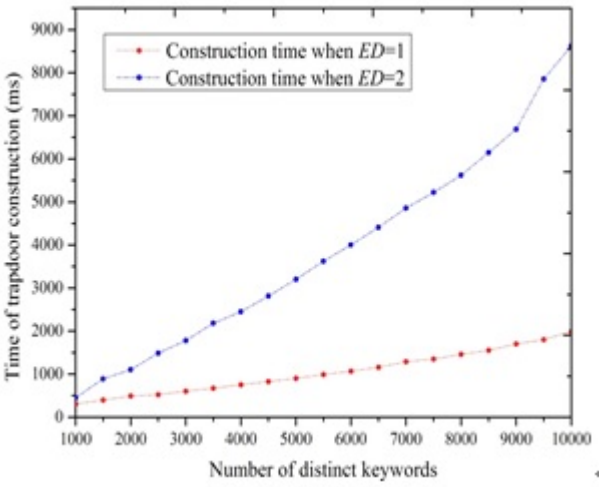

Figure 6: Time of trapdoor construction

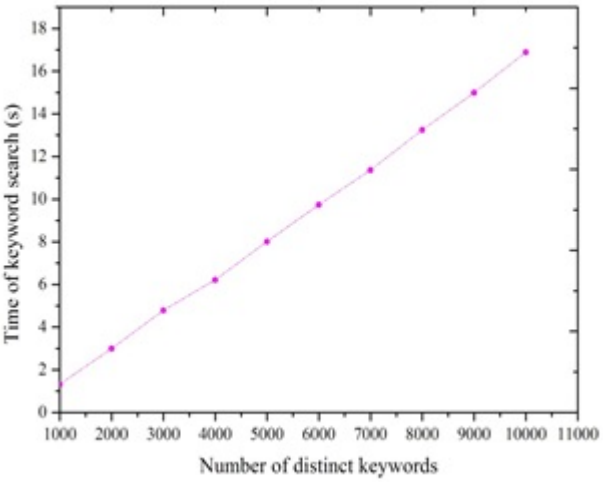

Figure 7: Time of keyword search

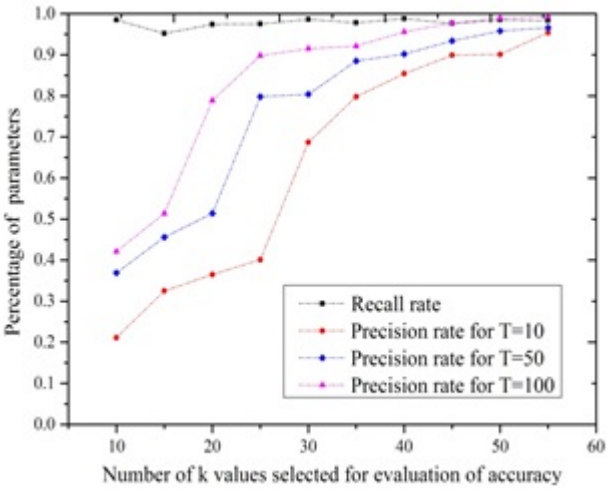

Figure 8: Performance evaluation

multi-keyword search on encrypted data and so on are likewise studied further. Application scenarios are greatly expanded due to different real work needs. Cao et al. [4], Wang et al. [17] proposed a ranked keyword search to protect privacy by using symmetric encryption schemes to achieve ranking. Different schemes are established to emphasize the keyword ranking of retrieval results in cloud encrypted data, which is the vital research direction in the searchable encryption field.

7 Conclusion

In this paper, we tackled the issue of fault-tolerant verifiable keyword search on cloud encrypted data. We considered the severe disadvantage of Li's scheme, which was found by Zheng presenting his conclusion in HPCC 2013 Conference, so it is important to construct a securely full-scale fuzzy keyword set so as to conduct fuzzy searching with verifiability in hybrid cloud. We proposed a novel scheme called **DFKSSVS**, which not only fully exploits infinite computing capability in the private cloud to accomplish fuzzy keyword set construction, but also supports verifiable-searching by *proof* sent from the public cloud, and dynamic ranking of retrieval results according to DU's searching history. Security analysis presents that our scheme can achieve provable-security of IND-CKA, because we have successfully converted fuzzy searching into exact keyword matching in the whole process which has been proven semantically secure before. Experimental results show accuracy, efficiency, and complexity of our new scheme, and compared with [5, 11, 19], our scheme can securely achieve our design goals—fuzzy matching, verifiable-searching, privacy-preservation, and retrieval accuracy based on dynamic ranking through feedback scheme.

Acknowledgments

This work is supported in part by Postdoctoral Research Foundation in Aviation University of Air Force. The authors would like to thank the Northeast Area Information Data Center Workshop discussion, and show great gratitude to the anonymous reviewers proposing constructive suggestions for this paper.

References

[1] J. Baek, R. Safavi-Naini, W. Susilo, "Public key encryption with keyword search revisited," in *Proceedings of International Conference on Computational Science and Its Applications (ICCSA'08)*, LNCS 5072, pp. 1249–1259, 2008.

[2] D. Boneh, G. D. Crescenzo, R. Ostrovsky, G. Persiano, "Public key encryption with keyword search," in *Proceedings of EUROCRYPT'04*, pp. 506–522, 2004.

[3] D. Boneh, B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proceedings of the 4th Theory of Cryptography Conference*, LNCS 4392, pp. 535–554, 2007.

[4] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, "Privacy preserving multi-keyword ranked search over encrypted cloud data," in *Proceedings of the 31th IEEE International Conference on Computer Communications (IEEE INFOCOM'11)*, pp. 829–837, 2011.

[5] Q. Chai, G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," University of Waterloo, 2011. (http://www.cacr.math.uwaterloo.ca/techreports/2011/cacr2011-22.pdf)

[6] M. Chuah, W. Hu, "Privacy-aware bed-tree based solution for fuzzy multi-keyword search over encrypted data," in *Proceedings of the 31st International Conference on Distributed Computing Systems Workshops (ICDCSW'11)*, pp. 273–281, 2011.

[7] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient construvtions," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 79–88, 2006.

[8] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," in *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC'09)*, pp. 169–178, 2009.

[9] E. Goh, "Secure indexes," Technical Report 2003/216, Cryptology ePrint Archive, 2003. (http://eprint.iacr.org/2003/216)

[10] P. Golle, J. Staddon, B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proceedings of Applied Cryptography and Network Security (ANCS'04)*, pp. 31–45, 2004.

[11] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proceedings of the 29th IEEE International Conference on Computer Communications (IEEE INFOCOM'10)*, pp. 441–445, 2010.

[12] C. Liu, L. Zhu, L. Li, 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 (IEEE CCIS'11)*, pp. 269–273, 2011.

[13] S. E. Ristad, N. Peter, "Learning string edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 522–532, 1998.

[14] D. Song, D. Wagner, A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P'00)*, pp. 44–55, 2000.

[15] W. Sun, B. Wang, N. Cao, M. Li, K. Ren, W. Lou, "Privacy preserving multi-keyword text search in the cloud computing supporting similarity-based ranking," in *Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security (ASIA CCS'13)*, pp. 71–82, 2013.

[16] B. Wang, S. Yu, W. Lou, T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proceedings of the 34th IEEE International Conference on Computer Communications (IEEE INFOCOM'14)*, pp. 1–9, 2014.

[17] C. Wang, N. Cao, J. Li, K. Ren, W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proceedings of the IEEE 30th International Conference on Distributed Computing Systems (ICDCS'10)*, pp. 253–262, 2010.

[18] C. Wang, K. Ren, S. Yu, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proceedings of the 32nd IEEE International Conference on Computer Communications (IEEE INFOCOM'12)*, pp. 451–459, 2012.

[19] J. Wang, X. Chen, H. Ma, Q. Tang, J. Li, H. Zhu, "A verifiable fuzzy keyword search scheme over encrypted data," *Journal of Internet Service and Information Security*, vol. 2, no. 1/2, pp. 49–58, 2012.

[20] M. Zheng, H. Zhou, "An efficient attack on a fuzzy keyword search scheme over encrypted data," in *Proceedings of 2013 IEEE International Conference on High Performance Computing and Communications (IEEE HPCC'13)*, pp. 1647–1650, 2013.

[21] W. Zhou, L. Liu, H. Jing, "K-gram based fuzzy keyword search over encrypted cloud computing," *Journal of Software Engineering and Applications*, vol. 6, no. 1, pp. 29–32, 2013.

Jie Wang received the MS degree in Aviation University of Air Force, China. His research interests include applied cryptography, secure cloud storage, SSE (Symmetric Searchable Encryption) as well as ASE (Asymmetric Searchable Encryption), and big-data mining. He has published more than 10 research papers in refereed domestic and international conferences and journals.

Xiao Yu is a post-doctor researcher at Aviation University of Air Force. He received his PhD degree in Chinese Academy in 2007. His research interests include information security and applied cryptography.

Ming Zhao is a professor at Aviation University of Air Force and received his doctor's degree in Jilin University in 2008. His current researches include cloud architecture, cloud storage and cloud computing security.