DOI: 10.2298/CSIS121104028W

# Efficient Verifiable Fuzzy Keyword Search over Encrypted Data in Cloud Computing

Jianfeng Wang[1], Hua Ma[1], Qiang Tang[2], Jin Li[3],
Hui Zhu[4,5], Siqi Ma[6], and Xiaofeng Chen[4]*

[1] Department of Mathematics, Xidian University, China
wjf01@163.com,ma_hua@126.com
[2] APSIA group, SnT, University of Luxembourg
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
qiang.tang@uni.lu
[3] School of Computer Science, Guangzhou University, China
jinli71@gmail.com
[4] State Key Laboratory of Integrated Service Networks,
Xidian University, China
xfchen@xidian.edu.cn
[5] Network and Data Security Key Laboratory of Sichuan
Provincezhuhui@xidian.edu.cn
[6] School of Computer Science and Technology,
Xidian University, China
xdmasiqi@hotmail.com

**Abstract.** As cloud computing becomes prevalent, more and more sensitive data is being centralized into the cloud by users. To maintain the confidentiality of sensitive user data against untrusted servers, the data should be encrypted before they are uploaded. However, this raises a new challenge for performing search over the encrypted data efficiently. Although the existing searchable encryption schemes allow a user to search the encrypted data with confidentiality, these solutions cannot support the verifiability of searching result. We argue that a cloud server may be selfish in order to save its computation ability or bandwidth. For example, it may execute only a fraction of the search and returns part of the searching result. In this paper, we propose a new verifiable fuzzy keyword search scheme based on the symbol-tree which not only supports the fuzzy keyword search, but also enjoys the verifiability of the searching result. Through rigorous security and efficiency analysis, we show that our proposed scheme is secure under the proposed model, while correctly and efficiently realizing the verifiable fuzzy keyword search. The extensive experimental results demonstrate the efficiency of the proposed scheme.

**Keywords:** searchable encryption, verifiable fuzzy search, cloud computing.

* The corresponding author: Xiaofeng Chen, xfchen@xidian.edu.cn

## 1. Introduction

As cloud computing becomes prevalent, storage outsourcing is widely used to reduce operational costs or private backups. By outsourcing their data in the cloud, data owners can obtain high quality data storage services, while reducing the burden of data storage and maintenance. To securely store the outsourced data on an untrusted cloud server, sensitive data should be encrypted before outsourcing [16], [18]. However, it is intractable for data owners to search the encrypted data in the server efficiently. The trivial solution of downloading the whole database and decrypting locally is clearly impractical, due to the huge amount of communication and computation cost [20]. Moreover, data owners may share their outsourced data with a large number of users. The individual users might want to only retrieve certain specific data files they interested in during a given session [15]. It is desirable to support the searching functionality on the server side, without decrypting the data and loss of data confidentiality. A popular method is searchable encryption, which can offer the user to selectively retrieve files through keyword-based search. In addition, the keyword privacy should be protected effectively since keyword usually contains important information of the data files.

Although various searchable encryption schemes have been proposed to perform search securely and effectively without decrypting the data files, it is assumed that the server is “honest-but-curious”. Specifically, the cloud server will follow our proposed protocol, but try to find out as much secret information as possible based on their possessions. However, we noticed that the cloud server may be selfish in order to save its computation ability or bandwidth, which is significantly beyond the conventional “honest-but-curious” server model. We consider a stronger adversary called “semi-honest-but-curious” server [9]. That is, the server may execute only a fraction of the search and returns part of the searching result honestly. Chai et al. firstly addressed this problem and proposed a verifiable keyword search scheme (VSSE) in [9]. In their solution, when the search behavior is completed, the server needs to prove to the user that the search result is correct and complete, which is named as `verifiable searchability`. However, the solution only supports the exact keyword search. In 2010, Li et al. [15] proposed a fuzzy keyword search scheme over encrypted data in cloud computing. However, they have not considered the issue of verifiable keyword search.

In this paper, we propose a new efficient verifiable fuzzy keyword search scheme, which not only supports verifiable fuzzy keyword search, but also reduces the verifying computation cost to $O(1)$. Specifically, our contribution can be summarized as follows:

- To the best of our knowledge, we propose the first verifiable fuzzy keyword search (VFKS) scheme, which not only enables fuzzy keyword search over encrypted data, but also maintains keyword privacy and the verifiability of the searching result.
- Through rigorous security analysis, our solution is secure and privacy preserving, while supporting the verifiability of the searching result.

- Our solution is highly efficient. For each query, the verifying computation cost is a constant complexity. Compared with the solution in [9], we reduce theverifying computation cost from $O(L)$ to $O(1)$, where $L$ is the length of the searched keyword.

The organization of this paper is as follows. The related works are analyzed in Section 2 . Some preliminaries are given in Section 3.The proposed verifiable fuzzy keyword search scheme is given in Section 4. The extension scheme in hybrid cloud is given in Section 5. The security and performance analysis is given in Section 6 and 7. Finally, conclusion will be made in Section 8.

## 2. Related Work

Recently, plaintext fuzzy keyword search solutions have been proposed [4], [14], [13]. These solutions are based on approximate string matching techniques, which allow user to search without using try-and-see approach for finding relevant information. At a first glance, it seems possible for one to directly apply these string matching algorithms to the context of searchable encryption by computing the trapdoors on a character base within an alphabet. However, this trivial construction suffers from the dictionary and statistics attacks and fails to achieve the search privacy.

Searchable encryption is a broad concept that deals with searches in encrypted data. The goal is to outsource encrypted data and be able to conditionally retrieve or query data without having to decrypt all the data [2]. Traditional searchable encryption schemes (SSE) [3], [5], [6], [7], [10], [11], [12], [19] have been proposed in recent years. Among those works, most are focused on efficiency improvements and security definition formalizations. The first practical Searchable encryption scheme in the symmetric setting was proposed by Song et al. [19] in 2000. In their solution, each word of a document is encrypted independently with a special two-layered encryption construct. Unfortunately, the scheme is not secure against statistical analysis across multiple queries and can leak the positions of the queried keywords in a document. The searching overhead is linear to the whole file collection length. To achieve more efficient search, Goh [12] proposed to use Bloom filters to construct the index for each file. The index makes the search scheme independent of the file encryption. Moreover, the complexity of each search request is roughly proportional to the number of files in the collection. Chang et al. [10] developed a similar per-file index scheme. Curtmola et al. [11] presented the formal security notion of searchable encryption. Furthermore, they proposed similar “index” approaches, where a single encrypted hash table index is built for the entire file collection. In the index table, each entry consisting of the trapdoor of a keyword and an encrypted set of related file identifiers. Bao et al. [3] proposed a searchable encryption scheme in multi-user setting, where a group of users share data in a way that can contribute searchable contents and can search an encrypted file collection without sharing their secrets.

Searchable encryption has also been studied in the asymmetric setting. The first public-key based searchable encryption scheme is presented by Boneh et al. [6] in 2004, where anyone with the public key can encrypt data but only authorized users with the private key are able to search. Subsequently, Abdalla et al. [1] proposed a novel public-key encryption with temporary keyword search. Compared to symmetric searchable encryption, public key solutions are usually very computationally expensive.

All existing secure index based schemes support only exact keyword search. Hence, such schemes are not suitable for cloud computing. Li et al. [15] proposed the first fuzzy keyword search over encrypted data in cloud computing, which utilized the multi-way tree to enhance the search efficiency. However, note that the semi-honest-but-curious cloud server may be selfish in order to save its computation ability or bandwidth. It may execute only a fraction of the search and returns part of the searching result honestly. To solve this problem, Chai et al. [9] proposed a verifiable SSE (VSSE) scheme, which ensures that the user can verify the correctness and completeness of the search result.

## 3. Preliminaries

### 3.1. Notions

$C = (F_1, F_2, \ldots, F_n)$: a set of $n$ encryption files;
$W = \{w_1, w_2, \ldots, w_p\}$: the set of distinct keywords of $C$;
$ID\{F_i\}$: the identifier of document $F_i$;
$ID_{w_i}$: the identifiers of documents containing the keyword $w_i$;
$F_{K,\cdot}$: a pseudo-random function; defined as $\{0,1\}^\star \times K \to \{0,1\}^l$;
$\{T_{w'}\}$: the trapdoor set of all fuzzy keywords of $w^{'} \in S_{w,d}$;
$\triangle = \{\alpha_i\}$: the predefined symbol set, where $|\triangle| = 2^n$, and $\alpha_i \in \triangle$ can be denoted by $n$ bits;
$G_W$: a tree covering all fuzzy keywords of $w \in W$ is built up based on symbols in $|\triangle|$;
$T_w[i]$: the i-th symbol of the symbol sequence of trapdoor $T_w$;
$ord(T_w[i])$: the alphabetic order of the character $T_w[i]$ in $\triangle$;

### 3.2. Definitions

A verifiable fuzzy keyword search scheme (VFKS) consists of the polynomial-time algorithms ($\mathbf{Keygen}$, $\mathbf{Buildindex}$, $\mathbf{Trapdoor}$, $\mathbf{Search}$), which are similar to those of standard symmetric searchable encryption scheme (SSE), as well as a new algorithm $\mathbf{Verify}$. These algorithms are defined as follows:

- $\mathbf{Keygen}(\lambda)$: This algorithm is run by the data owner to setup the scheme. It takes a security parameter $\lambda$ as input, and outputs the trapdoor generation key $sk$ and secret key $k$.

- $\mathbf{Buildindex}(sk, W)$: This algorithm is run by the data owner to create the index. It takes a secret $sk$ and the distinct keyword set $W$ of the document collection $C$ as inputs, and outputs a symbol-tree $G_W$.
- $\mathbf{Trapdoor}(sk, S_{w,d})$: This algorithm is run by the user to generate trapdoors for all fuzzy keywords of the user input keyword $w$. It takes a secret key $sk$ and a fuzzy keyword set $S_{w,d}$ as inputs, and outputs a trapdoor set $\{T_{w'}\}_{w' \in S_{w,d}}$.
- $\mathbf{Search}(G_W, \{T_{w'}\})$: This algorithm is run by the server in order to search for the files in $C$ that contain keyword $w$. It takes the symbol-tree $G_W$ of the file collection $C$ and a trapdoor set $\{T_{w'}\}$ of the fuzzy keyword set $S_{w,d}$ as inputs, and if search is successful outputs $ID_w$ and the $proof$, otherwise outputs the $proof$.
- $\mathbf{Verify}(k, proof)$: This algorithm is run by the user to test whether the server is honest. It takes a secret $k$ and $proof$ as inputs, and outputs $True$ if pass, otherwise outputs $False$.

**Edit Distance** Edit distance is a measure of similarity between two strings. The edit distance $ed(w_1, w_2)$ between two words $w_1$ and $w_2$ is the minimum number of operations required to transform one to the other. There are three primitive operations. (1) Substitution:changing one character to another in a word; (2) Deletion: deleting one character from a word; (3) Insertion: inserting a single character into a word. Given a keyword $w$, we let $S_{w,d}$ denote the set of keywords $w^{'}$ satisfying $ed(w, w^{'}) < d$ for a certain integer $d$.

**Trapdoors of Keywords** Trapdoors of the keywords are realized by applying a hash function $f$ as follows: Given a keyword $w$, we compute the trapdoor of $w$ as $T_w = f(sk, w)$, where the $sk$ is the user's index generation key.

**Verifiable of Keyword Search** The server executes search for the user when receiving the search request, and returns the search result and the $proof$. If the server executes all operations honestly, the probability that the search result is incorrect should be negligible; but if the server just returns a fraction of the search result honestly, the user can detect the cheating behavior with overwhelming probability through the $verify$ algorithm.

### 3.3. System Model

In this paper, we consider a cloud data-outsourcing system, which consists of three different entities: the data owner, the user and the cloud server. The data owner has a collection of $n$ encrypted data files $C = (F_1, F_2, \cdots, F_n)$ to be stored in the cloud server. A predefined set of distinct keywords in $C$ is denoted as $W = (w_1, w_2, \cdots, w_p)$. The cloud server performs fuzzy search for the authorized users over the encrypted data. We assume the authorization between the data owner and users is appropriately done. In the initialization phase, the data owner shares the trapdoor generation key $sk$ with authorized users, and builds an index $G_W$ for the encrypted file collection $C$ together with the encrypted files outsourcing to the cloud server. To search the file collection for any keyword $w$, an authorized user generates the trapdoor of $w$, and sends it to the cloud server. Upon receiving the search request by the user, the server performs the search over the index $G_W$ and returns all the encrypted files containing the specific keyword $w$. For the fuzzy keyword search, the server returns the closest possible results based on pre-specified similarity metrics. An architecture of verifiable fuzzy keyword search is shown in Fig. 1.

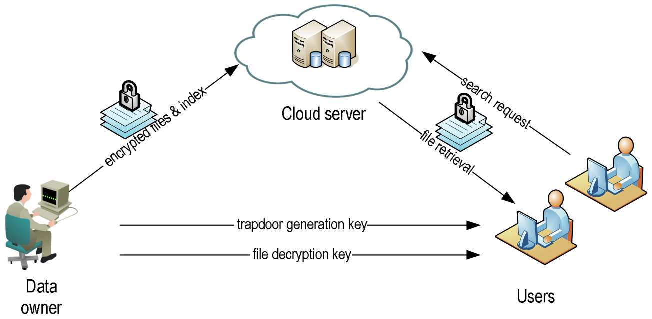


**Fig. 1.** Architecture of verifiable fuzzy keyword search

### 3.4. Security Model

In this work, we consider a “semi-honest-but-curious” cloud server, which is different with most previous searchable encryption schemes. We assume the cloud server acts in an “semi-honest” fashion, that is to say, it may not correctly follow our proposed protocol but forge part of search result or execute only a fraction of searching operations honestly. In addition, the cloud server tries to analyze the message flow received during the protocol in order to learn additional information. When designing verifiable fuzzy keyword search scheme, we follow the security definition deployed in the traditional searchable encryption [11]. Namely, it is required that noting should be leaked from the remotely stored files and index beyond the outcome and the pattern of search queries.

### 3.5. Design Goals

To support verifiable fuzzy keyword search over encrypted data using the above system and security models, our system design should achieve the following design goals: 1) to construct storage-efficient fuzzy keyword set and design efficient and effective fuzzy keyword search scheme; 2) to prevent the server from learning either the data files or the searched keywords beyond the search pattern and the access pattern; 3) to design efficient verifiable fuzzy keyword search scheme and enable user to verify the correctness and completeness of search result.

## 4. A New Verifiable Fuzzy Keyword Search Scheme

### 4.1. Construction of the VFKS scheme

In this section, we present the proposed scheme in detail. We assume the data files are separately encrypted by a symmetric cipher in a conventional manner before the user build the index. Our scheme consists of five algorithms $(\mathbf{KeyGen}, \mathbf{Buildindex}, \mathbf{Trapdoor}, \mathbf{Search}, \mathbf{Verify})$.

- $\mathbf{Keygen}$
  In this process, the data owner generates the index generation key $sk$ and a secret key $k$. The $\mathbf{Keygen}$ is a randomized key generation algorithm, which generate the key in this way : $sk, k \overset{R}{\leftarrow} \{0,1\}^k$.

**Algorithm 1** Generate Fuzzy Set ($w_i, d$)

```
Input: Keyword $w_i$ and Edit distance $d$
Output: Fuzzy keyword set $S_{w_i,d}$
if $d >= 1$ then
    Generate Fuzzy Set ($w_i, d - 1$);
end if
if $d = 0$ then
    Set $S_{w_i,d} = \{w_i\}$;
else
    for $k \leftarrow 1$ to $|S_{w_i,d}|$ do
        for $j \leftarrow 1$ to $2 \times |S_{w_i,d}[k]| + 1$ do
            if $j$ is odd then
                Set $Temp$ = $|S_{w_i,d}[k]|$;
                Insert $\star$ at position $j + 1/2$;
            else
                Set $Temp$ = $|S_{w_i,d}[k]|$;
                Replace $j/2$-th character with $\star$;
            end if
            if $Temp$ is not in $S_{w_i,d-1}$ then
                Set $S_{w_i,d} = S_{w_i,d} \cup \{Temp\}$;
            end if
        end for
    end for
end if
return $S_{w_i,d}$
```

- **Buildindex**
  In this process, we utilize a symbol-based trie-traverse search scheme, where a multi-way tree is constructed for storing the fuzzy keyword set $\{S_{w_i,d}\}_{w_i \in W}$ over a finite symbol set. The key idea behind this construction is that all trapdoors sharing a common prefix may have common nodes. The root is associated with an empty set and the symbols in a trapdoor can be recovered in a search from the root to the leaf that ends the trapdoor. All fuzzy keywords in the trie can be found by a depth-first search. Assume $\Delta = \{\alpha_i\}$ is a predefined symbol set, where the number of different symbols is $|\Delta| = 2^n$ and each symbol $\alpha_{_i} \in \Delta$ can be denoted by $n$ bits.

  1. **Initialization**
     - The data owner scans the $C$ and builds $W$, the set of distinct keywords of $D$.
     - The data owner outsources the encryption file collection $D$ to the server and receives the identifiers of each file ( denote as $ID\{F_i\}$ ). For all files of containing the keyword $w_i$, denote the identifier set as $ID_{w_i} = ID\{F_1\}\|ID\{F_2\}\ldots,\|ID\{F_i\}$.
  2. **Build Fuzzy Keyword Set**
     To build a storage-efficient fuzzy keyword set, we utilize the wildcard technique proposed in [15]. The idea is to consider the positions of the three primitive edit operations. Namely, we use a wildcard " $\star$ " to denote all edit operations at the same position. The wildcard-based fuzzy keyword set of $w_i$ with edit distance $d$ is denoted as $S_{w_i,d} = \{S^{'}_{w_i,0}, \cdots, S^{'}_{w_i,d}\}$, where $S^{'}_{w_i,d}$ denotes the set of keywords $w^{'}_i$ with $d$ wildcards. For example, for the keyword $cat$ with the pre-set edit distance 1, its wildcard-based fuzzy keyword set can be constructed as $S_{cat,1} = \{cat, \star cat, \star at, c\star at, c\star t, ca\star t, ca\star, cat\star\}$. The procedure for the wildcard fuzzy keyword set construction is shown in Algorithm 1.
  3. **Build Symbol-based Index Tree**
     - The data owner computes $T_{w^{'}_i} = f(sk, w^{'}_i)$ for each $w^{'}_i \in S_{w_i,d}(1 \leq i \leq p)$ with the index generation key $sk$. Then he divides the hash value into a sequence of symbols as $\alpha_{i_1}, \alpha_{i_2}, \cdots, \alpha_{i_{l/n}}$, where $l$ is the output length of one-way function $f(x)$.
     - The data owner builds up a trie $G_W$ covering all the fuzzy keywords $w_i \in W$. each node in $G_W$ has a two-tuples $(r_0, r_1)$, $r_0$ stores the symbol; $r_1$ stores a globally unique value $path||mem||F_k(path||mem)$, where $F_k(\cdot)$ is a pseudo random function. The $path$ contains a sequence of symbols from root to the current node and the $mem$ is a bitstream of length $2^n$, which represents the set of children of the current node. For example, if the current node is a child of root and has only one child whose $r_0$ is the i-th symbol in $\Delta$, the i-th bit of the bitstream of length $2^n$ is set to "1" while other bit positions are set to zero. That is, $path = \alpha_i$, $mem = 0\cdots1\cdots0$. The $r_1$ of leaf nodes is different to the other nodes, the $r_1$ can be denote as $r_1 = path||ID_{w_i}||F_K(path||ID_{w_i})$

  - The data owner attaches $\{ID_{w_i}|g_k(ID_{w_i})\}_{1\leq i\leq p}$ to $G_W$ and outsources $G_W$ with encrypted files to the cloud server.

- **Trapdoor**
  - For search input $w$, the user generates the fuzzy keyword set $S_{w,d}$ using the Algorithm 1;
  - For each $w^{'} \in S_{w,d}$, the user computes $T_{w'} = f(sk, w^{'})$ and sends $\{T_{w'}\}_{w'\in S_{w,d}}$ to the cloud server. Meanwhile, the user needs to temporary storage the $\{T_{w'}\}_{w'\in S_{w,d}}$, which is used during the verify process.

**Algorithm 2** Searching Tree $(G_W, \{T_{w'}\})$

```
Input: A trapdoor set $\{T_{w'}\}$ and The index tree $G_W$
Output: The set of proof $ProofSet$ and The set of files ID $IDSet$;
for $i \leftarrow 1$ to $|\{T_{w'}\}|$ do
    Set currentnode as root of $(G_W$;
    for $j \leftarrow 1$ to $l/n$ do
        Set $\alpha$ as $\alpha_{i_j}$ in the i-th $T_{w'}$;
        if no child of currentnode contains $\alpha$ then
            Append currentnode.proof to $ProofSet$;
            break;
        end if
        Set currentnode as child containing $\alpha$;
    end for
    if currentnode is leafnode then
        Append currentnode.proof to $ProofSet$;
        Append currentnode.ID to $IDSet$;
        if i = 1 then
            return $ProofSet$ and $IDSet$;
        end if
    end if
end for
return $ProofSet$ and $IDSet$;
```

- **Search**
  Upon receiving the search request, the server divides each $T_{w'}$ into a sequence of symbols, then performs the search over $G_W$ using Algorithm 2 and returns the file identifiers $ID_{w_i}$ and $proof$ to the user. According to the $ID_{w_i}$, the user can retrieve the files of his interest. Note that the $proof$ is the $r_1$ of each node, which is a globally unique value.
- **Verify**
  In this process, we introduce the method of verifying the searching result. The idea is that each node in $G_W$ has a globally unique value, called $proof$. Due to the construction of $G_W$, the path of each node is unique, without the secret key $k$, the attacker can not forge a valid $proof$. The data owner shares the $k$ with all authorized users. The authorized user can verify the correctness of the search result by reconstructing the $proof$.

- When the search is successful, firstly, the user utilizes the $IDSet$ to test the completeness of search result. Specifically, he computes $g_k(I\hat{D}_w)$ and tests whether $g_k(I\hat{D}_w)$ is equal to the received $g_k(ID_w)$, where $I\hat{D}_w$ is the concatenation of identifiers received by the user. If pass, then he utilizes the $ProofSet$ to test the correctness of search result. Similarly, the user computes $F_k(path|\hat{|}mem)$ and test whether $F_k(path|\hat{|}mem)$ is equal to the received $F_k(path||mem)$, where $path|\hat{|}mem$ is the former part of $r_1$ of the current node returned by the server. If $F_k(path|\hat{|}mem)$ is not equal to the received $F_k(path||mem)$, the user can defect that the server is not honest.
- When the search is not successful, the user directly tests the correctness of searching result. The process contains two steps:
  - $*$ The user tests whether $F_k(path|\hat{|}mem) = F_k(path||mem)$, if not equal, the user can defect that the server is not honest.
  - $*$ If step 1 pass, he tests whether $mem[ord(T_w[i+1])] = 1$, where the $T_w[i+1]$ is the next character of the current node in the symbol sequence of the trapdoor. If not equal, the user can defect that the server is not honest.

### 4.2. Performance Comparison

We compare the proposed algorithm with Li's scheme [15] and Chai's scheme [9]. To make the comparison easier, we assume that $N$ is the total number of keywords and $M$ is the maximum size of the fuzzy keyword set $S_{w_i,d}$. Table 1 presents the comparison of the search efficiency and the verifiability among the above schemes.

**Table 1.** Comparison of the three scheme

| | Li's scheme [15] | Chai's scheme [9] | Our scheme |
|---|---|---|---|
| Storage cost | $O(MN)$ | $O(N)$ | $O(MN)$ |
| Search cost | $O(1)$ | $O(L)$ | $O(1)$ |
| Verifiable searchability | No | Yes | Yes |
| Fuzzy searchability | Yes | No | Yes |
| Verify cost | - | $O(L)$ | $O(1)$ |

Compared with Li's scheme, our scheme achieve verifiable of search result. The verify cost shows the computation of verifying per trapdoor in each query. In Chai's scheme, it requires $L$ times decryption operations while in our scheme the computation is only one hash operation, where the $L$ is the length of the searched keyword. Note that our scheme can reduce the computation from $O(L)$ to $O(1)$, due to the query is performed frequently, we can reduce a large amount of computation at the user.

Though our scheme need more space to store the fuzzy keyword($O(MN)$), it achieves the the fuzzy searchability. All in all, our scheme not only supports the fuzzy search but also achieves the verifiable searchability more efficiently.

## 5. Verifiable Fuzzy Keyword Search in Hybrid Cloud

### 5.1. System and Security Model

Recently, Bugiel et al. [8] provided an architecture consisting of two clouds for secure outsourcing of data and arbitrary computations to an untrusted commodity cloud. Their approach consists of a private cloud and a public cloud. The private cloud performs the security-critical operations, whereas the public cloud performs the performance-critical ones. This allows maximum utilization of the expensive resources of the private Cloud, while high loads of queries can be processed on-demand by the public Cloud. Based on their two clouds architecture, we consider to address the privacy-preserving fuzzy keyword search problem simultaneously supporting verifiability of search result in hybrid cloud model.

In this section, we consider a extension from single cloud model to hybrid cloud model. There are four entities defined in the hybrid cloud model, that is, the data owner, the user, the private cloud and the public cloud. The data owner outsources the encrypted files to the public cloud and shares them with the authorized users. The user performs fuzzy keyword search and decrypts the encrypted files retrieved from the public cloud. The private cloud is additionally introduced to facilitate user's secure usage of cloud service. Specifically, since the computing resources at user side are restricted and the public cloud is not fully trusted in practice, the private cloud is able to provide users with an execution environment and infrastructure working as an interface between user and the public cloud. The interface offered by the private cloud allows user to securely submit files and queries to be securely stored and computed respectively. An architecture of verifiable fuzzy keyword search is shown in Fig.2.

In this model, we assume that the public cloud is “semi-honest-but-curious”, which may not correctly follow our proposed protocol but forge part of search result or execute only a fraction of searching operations honestly. As for the private cloud, we assume that it is “honest-but-curious”, which will follow our proposed protocol, but try to find out as much secret information as possible based on their possessions. In addition, the user's input keywords are allowed to be known by the private cloud. Actually, approximately relaxing security demands by allowing keywords leakage to private cloud is innocuous because the private cloud in practice is located in the premises of the organization [17].

### 5.2. Scheme Description

In single cloud model, data owner/user has to compute trapdoors for all the relevant fuzzy keywords for both index generation and search requesting, which

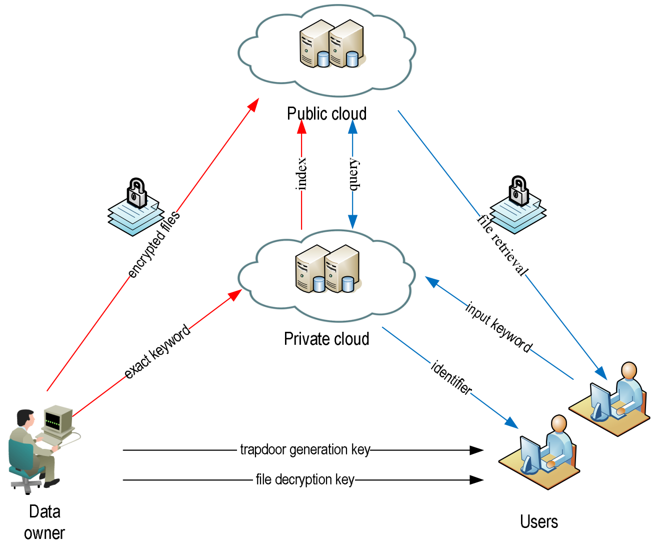


**Fig. 2.** Architecture of verifiable fuzzy keyword search

leads to a large amount of overhead at user side. In addition when receiving the search result retrieved from the public cloud, the user has to compute Pseudo-random function values for all the search results.

To fill the gap, we will show how to achieve efficient verifiable fuzzy keyword search under hybrid cloud model. The key idea is to outsource the expensive operation (i.e. trapdoor generation and search result verification) to the private cloud and only left the light-weight computation (file encryption and decryption) at user side.

Consider that the $Keygen$ operate identically to that in Section 4.1, we just provide the other four algorithms as follows.

- $\mathbf{BuildIndex}$ When outsourcing a file $F$, the data owner performs an encryption on $F$ by himself but outsources the task of generating the fuzzy keyword set $\{S_{w,d}\}_{w \in W}$ and building the index to the private cloud.
- $\mathbf{Trapdoor}$ and $\mathbf{Verify}$ When retrieving the interest files, the private cloud works as a proxy of the user. Firstly, it translates user's query into a set of trapdoors and sends to the public cloud. Later, upon receiving the search results returned by public cloud, the private cloud is to perform verification on them to test whether the public cloud is honest.
- $\mathbf{Search}$ Upon receiving the search request from the private cloud, the public cloud divides each trapdoor into a sequence of symbols, then performs the search and returns the search result to the private cloud.

Note that in hybrid cloud model, verifiable fuzzy keyword search can be presented soundly and efficiently. For soundness, since the private cloud possesses all the data owner/ user's resources except for the file encryption key $sk$, it can perform the operations of index generation and search result verification. Moreover, due to the private cloud performs all the overhead operations for the data owner/user, the whole process can be processed more efficiently.

## 6. Security Analysis

In this section, we prove the correctness and security of the proposed verifiable fuzzy keyword search scheme.

**Theorem 1.** *The intersection of the fuzzy sets $S_{w,d}$ and $S_{w_i,d}$ for keyword $w$ and $w_i$ is not empty if and only if $ed(w, w_i) \leq d$.*

**Proof.** First, we prove that $S_{w,d} \bigcap S_{w_i,d}$ is not empty if $ed(w, w_i) \leq d$. To prove this, it is enough to find out an element in $S_{w,d} \bigcap S_{w_i,d}$. According to the definition of edit distance, we can transform $w$ to $w_i$ after $ed(w, w_i)$ edit operations. From $w$, we get an element $w^\star$ by marking the positions of those $ed(w, w_i)$ operations on $w$ as $\star$. From $w^\star$, we can perform the $ed(w, w_i)$ edit operations on the same positions containing $\star$ at $w^\star$ and transform $w^\star$ to $w_i$. Since $ed(w, w_i) \leq d$, $w^\star$ is an element in both $S_{w,d}$ and $S_{w_i,d}$.

Next, we prove that $ed(w, w_i) \leq d$ if $S_{w,d} \bigcap S_{w_i,d}$ is not empty. We use $w^\star$ to denote the common element in $S_{w,d} \bigcap S_{w_i,d}$. Assume the number of $\star$ in $w^\star$ is $k$, there are two cases should be considered: If $k = 0$, it means that we do not need any edit operation to transform $w$ to $w_i$. That is, $w = w_i = w^\star$. Obviously, $ed(w, w_i) = 0 \leq d$. If $k > 0$, for any $\star$ in $w^\star$, we can perform edit operation on the position of the $\star$ and transform it to the corresponding character in $w$ and $w_i$. We use $w_1^\star$ and $w_{i_1}^\star$ to denote the result variants, respectively. Due to the two variants only have at most one position is different, we can transform $w_1^\star$ to $w_{i_1}^\star$ by at most one edit operation. That is, $ed(w_1^\star, w_{i_1}^\star) \leq 1$. After all the $k$ $\star$ be performed in $w^\star$, we get $w$ and $w_i$, respectively. Due to $w^\star \in S_{w,d} \bigcap S_{w_i,d}$, the number of $\star$ in $w^\star$ is not greater than $d$. we get that $ed(w, w_i) = k \leq d$.

**Theorem 2.** *The verifiable fuzzy keyword search scheme is secure regarding the search privacy.*

**Proof.** Similar to [15], suppose the proposed scheme is not achieve the index privacy against the indistinguishability under the chosen keyword attack (IND-CPA), which means there exists an algorithm $\mathcal{A}$ who can get the underlying information of keyword from the index. Then, we build an algorithm $\mathcal{A}'$ that utilizes $\mathcal{A}$ to determine whether some function $f'(\cdot)$ is a pseudo-random function such that $f'(\cdot)$ is equal to $f(sk, \cdot)$ or a random function. $\mathcal{A}'$ has an access to an oracle $O_{f'(\cdot)}$ that takes as input secret value $x$ and return $f'(x)$. Upon receiving any request of the index computation, $\mathcal{A}'$ answers it with request to the oracle $O_{f'(\cdot)}$. After making these trapdoor queries, the adversary outputs two keywords $w_0^*$ and $w_1^*$ with the same length and edit distance, which can be relaxed by adding some redundant trapdoors. $\mathcal{A}'$ picks one random $b \in \{0, 1\}$

and sends $w_b^*$ to the challenger. Then, $\mathcal{A}^{'}$ is given a challenge value $y$, which is either computed from a pseudo-random function $f(sk,\cdot)$ or a random function. $\mathcal{A}^{'}$ sends $y$ back to $\mathcal{A}$, who answers with $b^{'} \in \{0,1\}$. Suppose $\mathcal{A}$ guesses $b$ correctly with nonnegligible probability, which indicates that the value is not randomly computed. Then, $\mathcal{A}^{'}$ makes a decision that $f^{'}(\cdot)$ is a pseudo-random function. As a result, based on the assumption of the indistinguishability of the pseudo-random function from some real random function, $\mathcal{A}$ at best guesses $b$ correctly with approximate probability 1/2. Thus, the search privacy is obtained.

**Theorem 3.** *The verifiable fuzzy keyword search scheme is secure based on the verifiable fuzzy search.*

**Proof.** To prove the verifiability, we need to prove that the attacker can not forge a valid $proof$. To tamper the search result, the attacker need to forge the $proof$. There are three ways: (1) generate a $r_1$ with different parameter $path||mem$; (2) randomly generate a $r_1$ to replace the original one; (3) return the $r_1$ of another node back to the user.

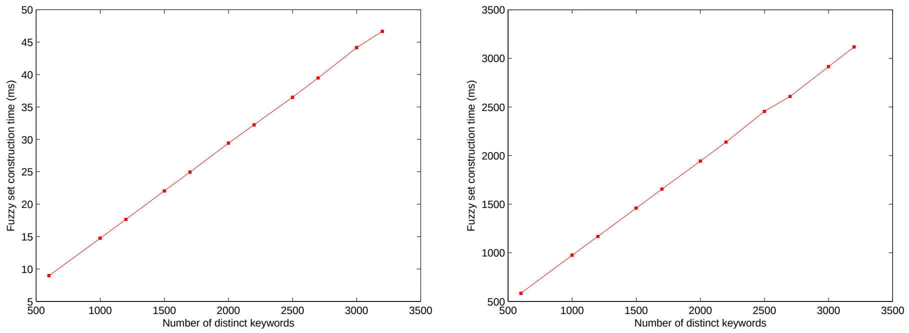
- For method (1) and (2), due to the collision resistance properties of hash function, each node in $G_W$ has a unique $r_1$, the attacker can successful cheat with a negligible probability without the secret key $k$. That is, the attacker can not return part of the search result or some fault one.
- For method (3), According to the construction of $G_W$, there has a unique $path$ from root to the current node. In other words, the $path$ of any node can be called signature of the node. The $r_1$ with the different $path$ will be reject by the algorithm $verify$.

Baesd on the above analysis, without the secret key $k$, the attacker can not construct a valid $prooof$. That is, our proposed scheme is secure based on the assumption of collision resistance of hash function.

## 7. Performance Analysis

In this section, we analyze the efficiency of the proposed scheme based on simulation. Since the process of file encryption is independent to the process of index construction, we focus on the symbol-tree based search algorithm. Our experiment is simulated on a LINUX machine with Intel Pentium Dual Core E5800 3.2GHz and 2G memory.

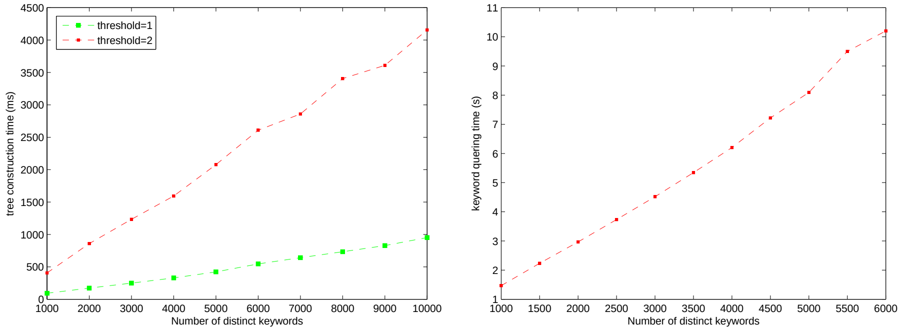
**Performance of Generating Fuzzy Keyword Set** In our experiment, we focus on the wildcard-based fuzzy set construction for all the keywords extracted from the file collection. Fig. 3 shows the fuzzy keyword set construction time with edit distance $d$=1 and 2. We can see that in both cases, the construction time increases linearly with the number of keywords. The cost constructing fuzzy keyword set under $d$=1 is much less than the case of $d$=2 due to the smaller set of possible wildcard- based words.



(a)

(b)

**Fig. 3.** Fuzzy keyword set construction time using wild-based approach:(a) $d$=1, (b) $d$=2.



(a)

(b)

**Fig. 4.** The evaluation of symbol-based index tree :(a) construction time (b) searching time

**Performance of Building Symbol-based Index** Given the fuzzy keyword set constructed using wildcard-based approach, we measure the time cost of symbol-based Index construction. Fig. 4(a) shows the time cost of building the symbol-based index tree in the case edit distance $d$=1 and 2. Although the time cost is not very low,the index construction process can be conducted off-line, it will not affect the searching efficiency. Fig. 4(b) shows the time cost of a single keyword query.

## 8. Conclusion

In this paper, we investigated the fuzzy keyword search problem in the scenario of a semi-honest-but-curious server, which may execute only a fraction of the search and return part of the searching result honestly. We proposed a new efficient verifiable fuzzy keyword search scheme, which not only supports fuzzy keyword search over encrypted data, but also enjoys the verifiability of the searching result. Though rigorous security and efficiency analysis, we showed that our method is secure and privacy-preserving, while correctly realizing the verifiable fuzzy keyword search.

**Acknowledgments.** This work is supported by the National Natural Science Foundation of China (No. 61272455 and 61100224), major national science and technology projects (No. 2012ZX03002003), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (No. 2011JQ8042) and the Fundamental Research Funds for the Central Universities (Nos. JY10000901034 and K50510010030).

## References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In: Proceedings of Advances in Cryptology - CRYPTO 2005. pp. 205–222 (2005)
2. Agudo, I., Nuñez, D., Giammatteo, G., Rizomiliotis, P., Lambrinoudakis, C.: Cryptography goes to the cloud. In: Lee, C., Seigneur, J.M., Park, J., Wagner, R. (eds.) Secure and Trust Computing, Data Management, and Applications, Communications in Computer and Information Science, vol. 187, pp. 190–197. Springer (2011)
3. Bao, F., Deng, R.H., Ding, X., Yang, Y.: Private query on encrypted data in multi-user settings. In: Proceedings of the 8th International Conference on Information Security Practice and Experience. pp. 71–85. Springer, Sydney, Australia (2008)
4. Behm, A., Ji, S., Li, C., Lu, J.: Space-constrained gram-based indexing for efficient approximate string search. In: Proceedings of the 25th IEEE International Conference on Data Engineering. pp. 604–615. IEEE, Shanghai, China (2009)
5. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Proceedings of Advances in Cryptology - CRYPTO 2007. pp. 535–552. Springer, Santa Barbara, CA, USA (2007)
6. Boneh, D., Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Proceedings of Advances in Cryptology - EUROCRYPT 2004. pp. 506–522. Springer, Interlaken, Switzerland (2004)

7. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Proceedings of the 4th Theory of Cryptography Conference. vol. 4392, pp. 535–554. Springer, Amsterdam, The Netherlands (2007)
8. Bugiel, S., Nurnberger, S., Sadeghi, A.R., Schneide, T.: Twin clouds: An architecture for secure cloud computing. In: Proceedings of the Workshop on Cryptography and Security in Clouds(WCSC'11). Zurich, Switzerland (2011)
9. Chai, Q., Gong, G.: Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. CACR, University of Waterloo (2011), [Online]. Available: http://www.cacr.math.uwaterloo.ca/techreports/2011/cacr2011-22.pdf (current December 2012)
10. Chang, Y., Mitzenmacher, M.: Privacy preserving keyword searches on remote encypted data. In: Proceedings of the 3rd Applied Cryptography and Network Security. pp. 391–421. New York, NY, USA (2005)
11. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definition and efficient constrcutions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. pp. 79–88. ACM, Alexandria, Virginia, USA (2006)
12. Goh, E.: Secure indexes. Report 2003/216, Cryptology ePrint Archive (2003), http://eprint.iacr.org/2003/216
13. Ji, S., Li, G., Li, C., Feng, J.: Efficient interactive fuzzy keyword search. In: Proceedings of 18th International World Wide Web Conference. ACM, Madrid, Spain (2009)
14. Li, C., Lu, J., Lu, Y.: Efficient merging and filtering algorithms for approximate string searches. In: Proceedings of the 24th IEEE International Conference on Data Engineering. pp. 257–266. IEEE, Cancun, Mexico (2008)
15. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K.and Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: Proceedings of the 29th IEEE International Conference on Computer Communications(INFOCOM'10). pp. 441–445. IEEE, San Diego, CA, USA (2010)
16. Lu, Y., Tsudik, G.: Privacy-preserving cloud database querying. Journal of Internet Services and Information Security 1(4), 5–25 (2011)
17. Poisel, R., Tjoa, S.: Discussion on the challenges and opportunities of cloud forensics. In: Proceedings of the International Cross-Domain Conference and Workshop on Availability, Reliability, and Security(CD-ARES'12). pp. 593–608. Springer, Prague, Czech Republic (2012)
18. Shiraishi, Y., Mohri, M., Fukuta, Y.: A server-aided computation protocol revisited for con-fidentiality of cloud servic. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications 2(2), 83–94 (2011)
19. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy. pp. 44–55. IEEE, Berkeley, California, USA (2000)
20. Wang, C., Ren, K., Yu, S., Mahendra, K.: Achieving usable and privacy-assured similarity search over out-sourced cloud data. In: Proceedings of the 31th IEEE International Conference on Computer Communications(INFOCOM'12). pp. 451–459. IEEE, Orlando, USA (2012)

**Jianfeng Wang** is a graduate student at the Faculty of Science, in Xidian University. His research interests include public key cryptography, cloud computing security and searchable encryption.

**Hua Ma** received her Masters degree in Applied Mathematics from the Xidian University in 1990. She is currently a Professor at the Faculty of Science, Xidian University. Her research interests include public key cryptography, network security, and electronic commerce.

**Qiang Tang** is a postdoc researcher in the Interdisciplinary Centre for Security, Reliability and Trust at University of Luxembourg. Before this, he was a postdoc researcher at the Distributed and Embedded Security Research Group in the Computer Science department at University of Twente, the Netherlands. He received his MSc degree from Peking University , China in 2002 and obtained his PhD degree from Royal Holloway, University of London , UK in 2007.

**Jin Li** He received his B.S. (2002) and M.S. (2004) from Southwest University and Sun Yat-sen University, both in Mathematics. He got his Ph.D degree in information security from Sun Yat-sen University at 2007. Currently, he works at Guangzhou University. His research interests include Applied Cryptography and Security in Cloud Computing (secure outsourcing computation and cloud storage). He served as a senior research associate at Korea Advanced Institute of Technology (Korea) and Illinois Institute of Technology (U.S.A.) from 2008 to 2010, respectively. He has published more than 40 papers in international conferences and journals, including IEEE INFOCOM, IEEE Transaction on Parallel and Distributed Computation, IEEE Transaction on Information Forensics and Security, ESORICS etc. He also served as TPC committee for many international conferences on security. He received a National Science Foundation of China (NSFC) Grant for his research on secure outsourcing computation in cloud computing. He was selected as one of science and technology new stars in Guangdong province.

**Hui Zhu** associate professor, born in 1981, received the Ph.D. degree in information security from Xidian University in 2009. His current research interests include network security and security authentication protocol.

**Siqi Ma** is a student at the School of Computer Science and Technology, in Xidian University. Her research interests include cloud computing security and network Security.

**Xiaofeng Chen** received his Ph.D. in cryptography from the Xidian University in 2003. He is currently a Professor at the School of Telecommunications Enginnering, Xidian University. His research interests include public key cryptography, financial cryptography, and cloud computing security.

*Received: November 4, 2012; Accepted: April 1, 2013.*