Received: 2021-03-12; Revised: 2021-03-28 doi:10.52396/JUST-2021-0071

An efficient multi-user multi-keyword fuzzy search scheme over encrypted cloud storage

LING Huaze[1], XUE Kaiping[1*], David S. L. WEI[2], LI Ruidong[3]

1. School of Cyber Security, University of Science and Technology of China, Hefei, Anhui 230027, China;
2. Computer and Information Science Department, Fordham University, NY 10458, USA;
3. College of Science and Engineering, Kanazawa University, Kakuma-machi, Kanazawa 920-1192, Japan
* Corresponding author. E-mail: kpxue@ustc.edu.cn

Abstract: As more and more enterprises and individuals choose to outsource their encrypted private data to the cloud, Searchable Encryption (SE), which solves the issue of keyword-searching over encrypted data, is becoming much more important. To overcome typos and semantic diversity existing in query requests, fuzzy search is introduced to achieve a misspelling-tolerate search-supported encryption scheme. However, current schemes of fuzzy search over encrypted data not only bring in high computing and communication overhead in multi-user scenarios but also are unable to cover all kinds of error types under the premise of an effective accuracy. In this paper, we thus propose a multi-user multi-keyword fuzzy searchable encryption scheme. Specifically, we introduce the permuterm index to support multi-keyword wildcard search which can solve more kinds of misspelling with a higher degree of correctness. Moreover, by letting the cloud server re-encrypt indexes user encrypt, our scheme supports unshared-key multi-user fuzzy search, reducing users´ computing overhead effectively and improving the level of privacy-preserving. The results of experiments demonstrate that, compared with existing schemes, our scheme not only has a better accuracy rate, but also supports more varieties of misspelling keyword search with acceptable computational overhead.

Keywords: encrypted cloud storage; proxy re-encryption; privacy preservation; searchable encryption
CLC number: TU459 **Document code**: A

1 Introduction

With the advent of the era of cloud computing, more and more enterprises and individuals choose to outsource the storage and management of their data to reduce their local management costs. However, private data must be encrypted before uploaded to avoid data leakage and abuse[1,2]. Meanwhile, essential data management including efficient keyword searching is necessary. In such a computing environment where the encrypted data have no semantics and no logic, it would be challenging to do content-related searches. A simple solution is to download all files and decrypt them locally, transforming the search over encrypted files to plaintext searching. However, such a solution makes outsourcing data to the cloud pointless and brings huge overhead. Therefore, we expect a solution to do searching directly over ciphertext possible.

Searchable Encryption (SE) is such a solution proposed to do a content-related query over ciphertext, which usually has three steps: index generation, encryption, and matching of encrypted index and query. Once a data user sends an encrypted query, the cloud server should be able to return the corresponding files, of which the encrypted index exactly match the encrypted query, without knowing any information. During the process, cloud service should be able to provide search services with the same level of querying quality, usability, and flexibility as plaintext search, and at the same time it should also guarantee the privacy protection for users' uploaded sensitive data.

However, errors in the keywords are unavoidable in the real world, which may make SE not return correct results. Thus fuzzy search[3,4] is proposed to achieve search tasks with misspelled or partially spelled query keywords. In fuzzy search, keywords are decomposed into several substrings through keyword transformation, and will be stored in bloom filters. Finally, the cloud server calculates the Euclidean distance between bloom filters of indexes' substrings and queries' substrings.

Citation: LING Huaze, XUE Kaiping, David S. L. WEI, et al. An efficient multi-user multi-keyword fuzzy search scheme over encrypted cloud storage. J. Univ. Sci. Tech. China, 2021, 51(7): 562−576.

And the files, of which the indexes have the least Euclidean distance will be returned.

Although many schemes have been proposed in terms of fuzzy searchable encryption[3-6], they still have many problems. Firstly, different methods of keyword transformation greatly affect the correctness and efficiency of the searching. Due to deficiencies in the methods to transform keywords, the accuracy rate of existing schemes is still low. Secondly, many schemes adopt locality-sensitive hashing (LSH) functions from the same hash family to generate the bloom filter. Benefiting from the feature of LSH functions that similar substrings (Euclidean distance between them is within a threshold) are mapped to the same positions of bloom filter with a high probability, these schemes can achieve matching of error keywords. However, once the error degree of keywords is higher than the threshold set by LSH, the accuracy rate of the fuzzy search will significantly decrease, which makes it difficult to support the multi-keyword fuzzy search scenario. Besides, the methods of keyword transform in existing schemes cannot support wildcard, which causes they cannot achieve the fuzzy search with more kinds of misspellings. Therefore, a more effective misspelling-tolerate method of keyword transformation is needed.

Moreover, to make fuzzy search schemes more practical, they should be efficient and effective in a multi-user scenario, where data users can query multiple repositories owned by different owners, meanwhile, a repository can be searched for multiple users. Some searchable encryption schemes in multi-user scenarios have already been proposed[7-9]. But unfortunately, as far as we know, there are no schemes that can support fuzzy search efficiently. Secure k-nearest neighbor (secure KNN), homomorphic encryption and attributed based encryption (ABE) are widely used in the multi-user searchable encryption. The schemes based on the first two algorithms often bring huge computational overhead, since when users make queries in different repositories, he/she must encrypt his/her query several times with the corresponding secret key of each data owner. Although the ABE-based schemes can decrease the times of encryption, it can not support fuzzy search. Thus, straightforwardly applying these schemes to the fuzzy search scenario is unfeasible and we need to explore a new solution to achieve multi-user fuzzy searchable encryption considering both searchable encryption and keyword transformation method.

Motivated by these observations, we propose an efficient multi-user multi-keyword fuzzy search scheme over encrypted cloud storage. Specifically, we develop a novel method of keyword transformation by introducing permuterm index, which can tolerate a higher degree of misspelling and support wildcard query. Meanwhile, in order to avoid the sharp decrease of accuracy rate caused by the increase of fuzzy keywords, our scheme stores the substrings in bloom filter directly and distinguishes substrings by using keyword length as a mark. And we apply an improved secure KNN where the index/query will be encrypted by users and re-encrypted by the cloud server in our scheme. Users can encrypt original indexes or queries with their keys for only one time, which greatly reduces the computing overhead. The main contributions of our work are summarized as follows:

(Ⅰ) We propose an efficient fuzzy search scheme over encrypted cloud storage in multi-user multi-keyword scenario. By applying an improved secure KNN where cloud server will re-encrypt owners' data and users' queries, our scheme largely reduce the computational overhead of users in multi-user scenario.

(Ⅱ) We also design a novel data transformation method by introducing permuterm index to make our scheme have a high degree of error tolerance and support wildcard query. Besides, in order to avoid the sharp decrease of accuracy caused by the increase in the number of fuzzy keywords in multi-keyword scenario, our scheme stores the substrings to bloom filter directly and distinguishes substrings by using keyword length as a mark.

(Ⅲ) We analyze the security strenth of our scheme, and evaluate the performance by conducting experiments, which shows that our scheme has higher accuracy, higher error tolerance, and less computational overhead while ensuring confidentiality, indistinguishability of query and privacy preservation.

The remainder of this paper is organized as follows. In Section 2, we introduce the system model, security assumption and design goals of our work. Preliminaries including bloom filter, permuterm index and secure KNN are introduced in Section 3. And the details of our multi-user multi-keyword fuzzy searchable encryption scheme are shown in Section 4. Then, we present the security analysis and performance evaluation in Sections 5 and 6 respectively. Then, the related work is given in the Section 7. Finally, we conclude our work in Section 8.

2 Problem statement

2.1 System model

Our system model consists of four main entities, namely trusted authority (TA), data owner (DO), data user (DU), and cloud service provider (CSP). Figure 1 illustrates the construction of our system.

(Ⅰ) TA is responsible for generating, distributing and managing system parameters, and it is trusted by all entities in the system. During the system initialization, TA generates pairs of secret key and switch key

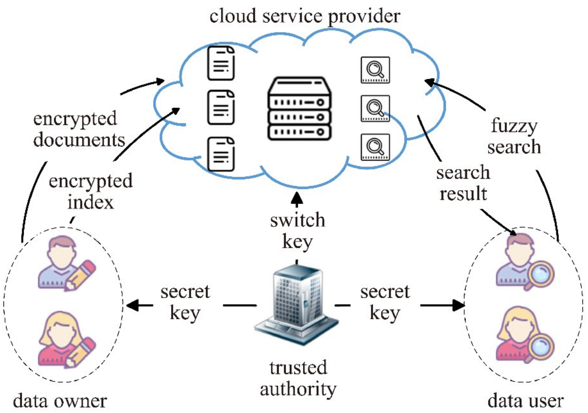


Figure 1. System Model

according to users' privileges, then distributes them to users and the cloud server respectively.

(Ⅱ) DO has a collection of files, and prefers to upload them to the cloud server sharing with other users. Before uploading the files, DO needs to build an encrypted index based on a set of keywords, which will also be uploaded to the cloud sever.

(Ⅲ) DUs make requests to the CSP by sending encrypted query, expecting the data files which match the keywords in his/hey query.

(Ⅳ) CSP provides the services of data storing, querying, and computing for DO and DU. Upon receiving the query from DU, CSP conducts keyword search and returns the matched files according to the keywords in the query.

2.2 Security assumption

Following References [7,8], we assume the CSP is "honest-but-curious"'. In this model, the CSP would fulfill its duties as a service provider by following the designated protocols and procedures honestly. At the same time, the CSP is curious and it may collect and analyze the data uploaded by users to obtain some additional information. In this paper, we consider two threat models Known Ciphertext Model and Known Background Model. They have been introduced in other related works[3,4].

(Ⅰ) Known ciphertext model. In this model, the CSP or adversaries has the full access rights to encrypted files, encrypted indexes, and encrypted queries uploaded by data users and data owners. Besides, the CSP or adversaries will record the results of each search request in an attempt to get useful information from them[10].

(Ⅱ) Known background model. The CSP or adversaries in this model can get additional background information, such as the similarities or differences between two given queries. Combined with background information, the CSP or adversaries can analyze the similarity between the target index and the known index, even can determine whether a certain keyword is in the index with a high probability. Specially, the CSP or adversaries cannot obtain the plaintext- ciphertext pair available for query as users' secret key are not available.

2.3 Design goals

In this paper, our scheme intends to achieve efficient fuzzy search in multi-user multi-keyword scenario. Therefore, our design has the following security and performance goals.

(Ⅰ) Multi-keyword fuzzy search with wildcard enable. Our multi-keyword fuzzy search scheme should support not only common misspelling but also wildcard search that tolerates more types of spelling errors. For example, the file which have a index information, secure should be returned for a wildcard query with keywords "se * re, informa * ".

(Ⅱ) No predefined dictionary. No predefined dictionary is a standard requirement of existing schemes, so our scheme should also not have a predefined dictionary.

(Ⅲ) High result accuracy. Our scheme is to provide users with keyword search function over ciphertext, so it is necessary to ensure high accuracy.

(Ⅳ) Less computational overhead in multi-user scenario. The computing overhead of users in our scheme should be significantly reduced, which means data user should encrypt an index for only one time when searching multiple repositories.

(Ⅴ) Privacy guarantee. The CSP or external adversary should be prevented from obtaining any additional information from encrypted indexes, encrypted queries or search results. Besides, the query information of different users should also be confidential to other users.

3 Preliminaries

3.1 Bloom filter

Bloom filter is proposed by Bloom in 1970[11], which is widely used to determine whether an element belongs to a collection. Bloom filter is initialized as a m-bit bit array A of with all bits set as 0. Given a set $S=\{a_1,a_2,\cdots,a_n\}$, bloom filter inserts each element $a_i \in S$ into the bit array. It uses l independent hash function: $H=\{h_j | h_i: S \to [1, m], 1 \leqslant j \leqslant l\}$ to set the $h_j(a_i)$-th bit in the array to 1. Testing whether a set S has the element q, q will be mapped to l positions by calculating $h_j(q)$, $1 \leqslant j \leqslant l$. If any of bit value corresponding to these l positions equals 0, it means $q \notin S$; otherwise, it means either $q \in S$ or q produces a false positive. Figure 2 is a simple example, where "x" and "$y=b$" can both be considered as an element in the set where "a", "b" belong to, but actually "x" is not such an element. For

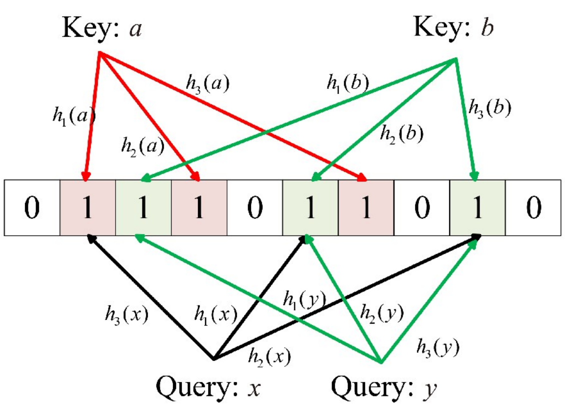


Figure 2. An example of bloom filter.

more details, References[12,13] are good references.

Following the analysis of Reference[11], the false positive probability f is

$$f = (1 - \{e^{-\frac{\ln}{m}})^{l} \qquad (1)$$

where n is the number of elements expected to be stored in the bloom filter, l is the number of hash functions used to implement the bloom filter, and m is the length of the bloom filter. Based on the excepted number of elements and false positive probability, we can choose the length of bloom filter m and number of hash functions l as Equation1.

3.2 Permuterm index

Permuterm index is a common index structure used to deal with wildcard query, such as the query with keywords "＊earch", "se＊rch", "searc＊". Here "＊" represents characters of any length. To generate the corresponding permuterm index for a keyword, a character "$" are added to the end of the keyword to present the ending. Then the keyword will be rotated bit by bit. For the keyword with a wildcard character "＊" in it, the permuterm index will be the string, of which the end is "＊". For the keyword without a wildcard character, the index will be the set of all the possible string generated by rotation. Now we use two specific examples to explain the establishment of permuterm index. Keyword "model" will be transformed into "model $", "odel $ m", "del $ mo", "el $ mod", "l $ mode", and keyword "mod ＊ l" will be transformed into "l $ mod＊".

3.3 Secure KNN

Secure KNN was proposed by Wong et al[14] solving the problem of KNN computation on an encrypted database. KNN algorithm is mainly used to search in a point set to find out k nearest points to a given point q. As the distance between two points can be computated as the product of their coordinate vector, secure KNN set the encryption secret key as a matrix and the decryption key as the inverse of the matrix, can be offset in distance comparison. Here is a brief introduction to the process of the secure KNN.

(Ⅰ) Key generation$(n)\rightarrow\kappa$. The key generation algorithm takes the length of the coordinate vector n as the input and returns a invertible matrix $M_{n\times n}$ as the secret key κ.

(Ⅱ) Coordinate vector encryption$(\kappa, P)\rightarrow E_{\kappa}(P)$. Given the coordinate vector of a point $P = p_{n\times 1}$ and the secret key κ, the encrypted point $Enc_{\kappa}(P) = M^{\mathrm{T}} \cdot p$ will be returned.

(Ⅲ) Query encryption$(\kappa, Q)\rightarrow \mathrm{Enc}_{\kappa}(Q)$. This algorithm takes a query point $Q = q_{n\times 1}$, which is also a n-dimensional vector, and the secret key κ as inputs. Then it outputs the encrypted point $E_{\kappa}(Q) = M^{-1} \cdot q$.

(Ⅳ) Distance comparison $(E_{\kappa}(P_1), E_{\kappa}(P_2), E_{\kappa}(Q))\rightarrow$Ture or False. Given encrypted points and query point $E_{\kappa}(P_1)$, $E_{\kappa}(P_2)$ and $E_{\kappa}(Q)$, the system calculate whether $(E_{\kappa}(P_1) - E_{\kappa}(P_2))^{\mathrm{T}} \cdot E_{\kappa}(Q) > 0$ to determine whether P_1 is nearer to the query Q than P_2.

3.4 Proxy re-encryption

Proxy re-encryption is an encryption technology that can transform a ciphertext encrypted by Alice to the ciphertext under Bob's key without decrypting. It was proposed by Blaze et al[15] and formally defined by Ateniese et al[16]. In a general proxy re-encryption scheme, there are five polynomial time algorithms KenGen, ReKey, Encrypt, ReEncrypt, Decrypt, here is a brief introduction to the process of the Proxy re-encryption.

(Ⅰ) KeyGen$(1^{k})\rightarrow(pk_i, sk_i)$. Given the security parameter 1^{k}, the KeyGen algorithm takes a key pair (pk_i, sk_i) as the output.

(Ⅱ) ReKey $(pk_A, sk_A, pk_B, sk_B)\rightarrow rk_{A\rightarrow B}\}$. The ReKey algorithm takes two key pairs (pk_A, sk_A, pk_B, sk_B) as the input, and a re-encryption key $rk_{A\rightarrow B}$ will be returned.

(Ⅲ) Encrypt$(pk_i, m)\rightarrow c_i$. This algorithm takes user's public key pk_i, and a message $m \in M$ as inputs. Then it outputs the ciphertext c_i.

(Ⅳ) ReEncrypt $(c_A, rk_{A\rightarrow B})\rightarrow c_B$. Given a ciphertext c_A and corresponding re-encryption key $rk_{A\rightarrow B}$, the ReEncrypt algorithm will return c_B as output.

(Ⅴ) Decrypt$(sk_i, c_i)\rightarrow m$. Given user's secret key sk_i and ciphertext c_i, the Decrypt algorithm outputs the message m or a error symbol $\perp$ indicated that the ciphertext c_i is illegal.

4 Basic construction of our scheme

4.1 Overview

In our scheme, to achieve fuzzy search, all the keywords in both queries and indexes need to be transformed into permuterm indexes. The system starts with system intialization. Before DOs upload their files, they first generate corresponding index for each file, which consists of multiple keywords. After keyword transformation, DOs directly store the permuterm indexes to bloom filter, which can increase the accuracy rate in multi-keyword scenario. Then they encrypt and upload corresponding bloom filter together with files they own to CSP. In order to efficiently support the multi-user scenario, upon receiving the uploaded files and indexes, CSP will re-encrypt the bloom filters, and then store them to wait for querying. Similarly, when DUs want to access some files, they first generate related keywords, and then transform them into permuterm indexes, which are stored in a bloom filter. After DUs send encrypted bloom filter as query, CSP will re-encrypt this bloom filter, and return the files of which have the matching encrypted bloom filter.

We will introduce the details of our whole construction in the following subsections, which includes system initialization, keyword transformation, index generation, index encryption, index conversion, query generation and search.

4.2 System initialization

Assuming that each index and each query are both an m-dimension vector, TA first generates the secret key $SK= \mathscr{M},\mathscr{M}^{-1}$ randomly. Here $\mathscr{M} \in \mathbb{Z}_h^{2m*2m}$ is an invertible matrix, where h is an integer that defines the range of data values and m is the length of bloom filter we used as indexes/queries. Based on SK, TA splits $\mathscr{M},\mathscr{M}^{-1}$, generates the secret key and switch key for DO/DU. Specifically, for each DO, TA generates secret key $sk_O = \mathscr{M}_{O_\theta}$ and corresponding switch key $\mathscr{M}'_{O_\theta}$, satisfying $\mathscr{M} = \mathscr{M}_{O_\theta} \times \mathscr{M}'_{O_\theta}$. Similarly, TA generates secret key $sk_U = \mathscr{M}_{U_\theta}$ and corresponding switch key $\mathscr{M}'_{U_\theta}$, where $\mathscr{M}^{-1} = \mathscr{M}_{U_\theta} \times \mathscr{M}'_{U_\theta}$ for each DU. In the end, TA sends all the switch keys to CSP, and sends secret keys to corresponding DOs/DUs.

4.3 Keyword transformation

For a file set $F = \{f_1, f_2, \cdots, f_n\}$, DO first extracts keywords from each file. Then he/she transforms keywords into permuterm indexes. The strings in these indexes are finally split into the corresponding substrings, which will represent the index of each file by combining with the length of the original keywords. The details of the process is shown in line 2−11 of Algorithm 4.1. The rotate (s, i) in the algorithm represents the function to rotate string s for i bit.

Algorithm 4.1 Keyword transformation and index/query generation

```
Input: The set of keywords for a file K; l independent hash
       function $h_i(x)_{i=1}^{l}$; A m-bit bit array B with all positions
       set as 0;
Output: The index/query for a file B;
1 for each word in K do
2     L=len(word);
3     if ‘ * ’ in word then
4         word=replace( ‘ * ’, ‘’);
5         out=‘ $ ’+word+‘’+‘ $ ’;
6         gram=out. split();
7     else
8         s=word+‘ $ ’+L +‘’+‘ $ ’;
9         for i in range( len(s)−5, 0, −1) do
10            out=rotate(s, i) + L;
11            gram=out. split();
12    for each element in gram do
13        for i=1 to l do
14            seat=$h_1$(element);
15            B[seat]=1;
16 Return B.
```

As an example, the permuterm index for “fuzzy” is “fuzzy $ ”, “uzzy $f”, “zzy $fu”, “zy $fuz”, “y $fuzz”. Then, the final representation is {“fuzzy $5”, “uzzy $5”, “ $f5”, “zzy $5”, “ $fu5”, “zy $5”, “ $fuz5”, “y $5”, “ $fuzz5”}. In such a way, a wildcard keyword like “tur * ” or a misspelled keyword like “fuzsy” can still be represented in a unified way with an accurate keyword and matched with “ture” and “fuzzy”, as shown in Figure 3.

4.4 Index generation

As keywords of an index are transformed into a substring-based set, we use l independent hash functions, $H=\{h_{\mathrm{i}} \mid h_{\mathrm{i}}: S \rightarrow [1, m], 1 \leqslant i \leqslant l\}$, to hash each substring s in the set with the length L of the original keyword together. For each keyword, we will have l positions $h_i(s|L)$ and set corresponding positions in the m-bit bloom filter to 1, which is shown in line 12 −15 of Algorithm 1. In this way, each file has an bloom filter as the index, which is shown at the bottom of Figure 3.

4.5 Index encryption

The entire encryption process in our scheme can be divided into two parts, i. e., the encryption of index at owner side and the re-encryption at CSP side. The process enables the indexes encrypted different keys to converted into indexes encrypted by the same key, i. e., SK, as shown in Figure 4. Users first use their own keys to encrypt indexes as the process of encrypt in Figure 4, then CSP convert the ciphertext under user's key to the ciphertext under the unified key as the process of re-encrypt in Figure 4.

After generating the m-bit index based on the keywords of a file, which can be seen as a m-dimension vector $\tilde{I} = \{i_1, i_2, \cdots, i_m\}$, DO extends $\tilde{I}$ to (2m)-dimensional vector $\overrightarrow{I_{\mathrm{i}}}$ as

$$\overrightarrow{I_{\mathrm{i}}} = \left\{i_1, i_2, \cdots, i_m, -\frac{1}{2}\sum_{j=1}^{m} i_j^2, \alpha_1, \alpha_2, \cdots, \alpha_{m-1}\right\} \tag{2}$$

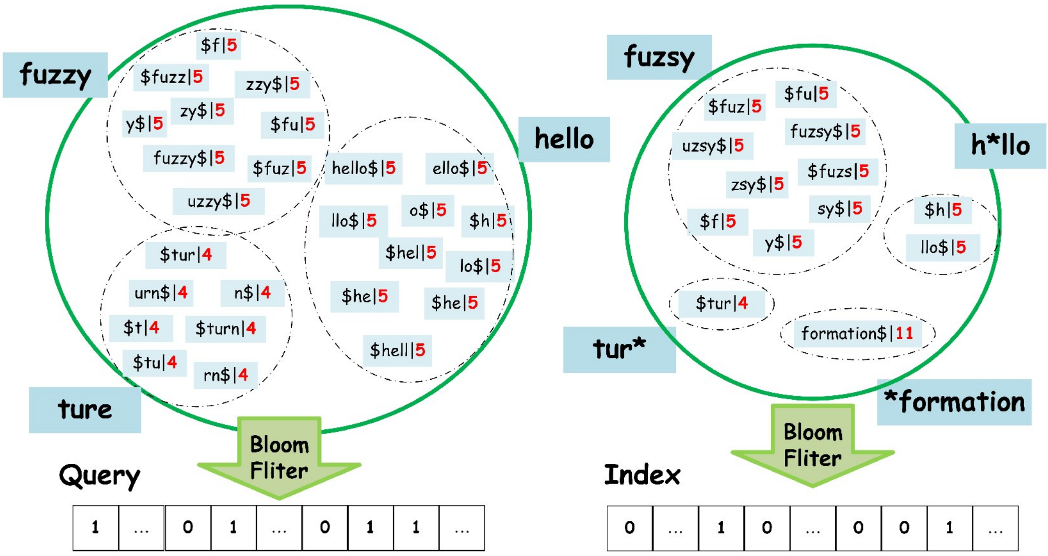


Figure 3. Keyword transformation and index/query generation.

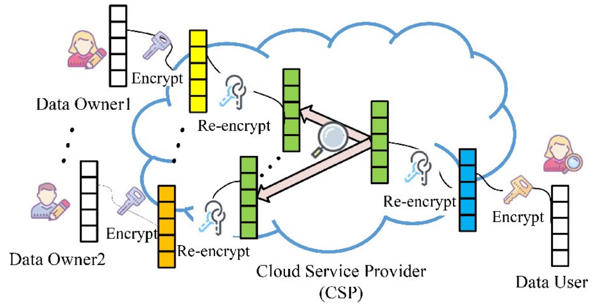


Figure 4. Encryption of Index.

where $\alpha_1, \alpha_2, \cdots, \alpha_{m-1} \in \mathbb{Z}_h$.

Then, DO encrypts the index using Equation (3), and outsources the encrypted index and encrypted file to CSP. The method to encrypt files is not in the scope of our paper.

$$E_{\mathscr{M}_O}(\overrightarrow{I_\mathrm{i}}) = \Gamma \cdot I_i + \overrightarrow{\varepsilon_i}) \times \mathscr{M}_O \qquad (3)$$

Here, $\Gamma \in \mathbb{Z}_l$, $l \geqslant \text{poly}(2m) \gg h \gg 2 \leqslant |\max \leqslant (\overrightarrow{\varepsilon_i})|$, $\overrightarrow{\varepsilon_i} \in \mathbb{Z}_l^{2m}$, is an integer error vector randomly selected from some probability distributions $\chi \in \mathbb{Z}_l$.

4.6 Index conversion

After obtaining encrypted $E_{\mathscr{M}_O}$, CSP re-encrypts it using Equation (4), making it encryted by $\mathscr{M}$. Then CSP store it.

$$E_{\mathscr{M}} = (\overrightarrow{I_i}) = E_{\mathscr{M}_O}(\overrightarrow{I_i}) \times \mathscr{M}'_O \qquad (4)$$

4.7 Query generation

When DU wants to make a query, DU first generates the keywords of the files he/she expects. Then he/she generates the query vector $\widetilde{Q} = \{q_1, q_2, \cdots, q_m\}$ according to the Algorithm 4.1. After that, DU conducts query encryption. Specifically, DU extends $\widetilde{Q}$ to $(2m)$-dimensional vector $\overrightarrow{Q}$ as

$$\overrightarrow{Q} = \{\gamma q_1, \gamma q_2, \cdots, \gamma q_m, \gamma, \beta_1, \beta_2, \cdots, \beta_{m-1}\} \qquad (5)$$

where $\beta_1, \beta_2, \cdots, \beta_{m-1} \in \mathbb{Z}_h$, and $\gamma \in \mathbb{Z}_p$ is a positive integer.

Subsequently, DU encrypts the query vector $\overrightarrow{Q}$ using Equation (6), and submits T_{Q_U} to CSP.

$$T_{Q_U} = \mathscr{M}_U \times (\Gamma \cdot \overrightarrow{Q}^\mathrm{T} + \overrightarrow{\varepsilon_q}^\mathrm{T}) \qquad (6)$$

where $\overrightarrow{\varepsilon_q}^\mathrm{T} \in Z_l^{2m}$ is a random integer error vector generated for each $\overrightarrow{Q}^\mathrm{T}$, and $\overrightarrow{Q}^\mathrm{T}$ is the column vector of $\overrightarrow{Q}$.

4.8 Search

Upon receiving the search query T_{Q_U} from DU, CSP first conducts query conversion. Specifically, it converts T_{Q_U} to be encrypted under $\mathscr{M}^{-1}$ by re−encrypting it with the corresponding switch key $\mathscr{M}_U'$ shown as

$$T_Q = \mathscr{M}'_U \times T_{Q_U} \qquad (7)$$

CSP sorts files based on the similarity between T_Q and each index $\{E_{\mathscr{M}}\}\overrightarrow{(I_i)}$. Here we use euclidean distance to measure the similarity. Finally, the list including top-k ranked files will be returned.

Correctness Guarantee: Provided two different encrypted queries, we can compare the similarity between these two indexes and the given query. Since we use the same bloom filter mechanism to generate these query and indexes, the same keyword information will be stored in the same set of each bloom filter. So the repetition of the keyword sets directly affects the distance between the bloom filters and the distance between two bloom filters, that is, the distance between two vectors, is commonly defined as Euclidean distance. So the correctness guarantee for searching holds as follow:

The Euclidean distance comparison is shown as Equation(8), where $E_{\mathscr{M}}\overrightarrow{(I_a)})$, $E_{\mathscr{M}}\overrightarrow{(I_b)})$ are the two different encrypted indexes used to do a comparison with the given query T_Q. The random numbers added when generating indexes and queries will be eliminated during the calculation and the coefficient $\frac{\gamma}{2}$ in the result is positive number. So when the result is greater than 0, we can say that $\widetilde{(I_b)}$ has a higher matching degree. And the fact that

$$\text{Comp}_{ab}\left[\frac{E_{\mathscr{M}}\overrightarrow{(I_a)})\ \times T_Q}{\Gamma^2}\right]\left[\frac{E_{\mathscr{M}}\overrightarrow{(I_b)}\ \times T_Q}{\Gamma^2}\right]=$$

$$\sum_{j=1}^{m}\gamma(q_j\cdot i_{aj}-\frac{1}{2}i_{aj}{}^2)+\sum_{j=1}^{m-1}\alpha_{aj}\beta_j-$$

$$\sum_{j=1}^{m}\gamma(q_j\cdot i_{aj}-\frac{1}{2}i_{bj}{}^2)-\sum_{j=1}^{m-1}\alpha_{bj}\beta_j+t$$

$$=-\frac{\gamma}{2}\left(Dis\{\widetilde{Q},\widetilde{I_a}-\sum_{j=1}^{m}q_j^2\right)+$$

$$\frac{\gamma}{2}\left(Dis\{\widetilde{Q},\widetilde{I_b}\}-\sum_{j=1}^{m}q_j^2\right)+\mu+t\approx$$

$$\frac{\gamma}{2}(Dis\{\widetilde{Q},\tilde{I}_b-Dis\{\widetilde{Q},\tilde{I}_a\})\tag{8}$$

where $\mu=\sum_{j=1}^{m-1}(\alpha_{aj}-\alpha_{bj})\beta_j$ can be ignored, and $t=\frac{(\vec{\varepsilon}_a-\vec{\varepsilon}_b)\times\vec{\varepsilon}_q^{\mathrm{T}}}{\Gamma^2}+\frac{(\vec{\varepsilon}_a-\vec{\varepsilon}_b)\times\vec{Q}_q^{\mathrm{T}}+(\vec{I}_a-\vec{I}_b)\times\vec{\varepsilon}_q^{\mathrm{T}}}{\Gamma}$ can also be ignored because of the fact that $\Gamma\gg 2\leqslant|\max(\vec{\varepsilon}_i)|$ and $\Gamma\gg 2\leqslant|\max(\vec{\varepsilon}_q)|$[17].

5 Security analysis

Inspired by Fu's work, we will analyze the security of our scheme and give a security proof in detail for known ciphertext model and known background model. Before the proof, we give some notations here.

(Ⅰ) History, $H=(F,\ I,\ Q)$. An interaction between the users and CSP can be defined as history. A history can be regarded as an instantiation of such an interaction, including a file set to be searched, searchable indexes and a set of words that DU wants to search for.

(Ⅱ) View, $V(H)=(\text{Enc}(F),\ \text{Enc}(I),\ \text{Enc}(I))$. Then during a query, what CSP or adversaries can actually obtain we call as the history's view. view is generated by encrypting a history with users' secret key. Additionally, CSP or adversaries can obtain some common information, such as the number of files stored in CSP.

(Ⅲ) Trace of a history, $\text{Tr}(H)=\text{Tr}(q_1)$, ⋯, $\text{Tr}(q_k)$. A trace of the history H is the set of the trace of queries $\text{Tr}(H)=\text{Tr}(q_1)$, ⋯, $\text{Tr}(q_k)$ consisting the sensitive information. Specifically, $\text{Tr}(q_i)=\{(\delta_{j,s_j})_{q_i\subset\delta_j},1\leqslant j\leqslant\leqslant|F|\}$, where s_j is the similarity score between the query q_i and the file δ_j.

5.1 In known ciphertext model

As introduced above, an index/query is a m-bit bloom filter and is extended to a $(2m)$-dimensional vector. Considering index an $\vec{I_i}$, it is encrypted by Equation(3) and converted by Equation(4). Both of $E_{\mathscr{M}O}(\vec{I_i})$ and $E_{\mathscr{M}}(\ \vec{I_i})$ can be obtained by CSP or adversaries in known ciphertext model. In known ciphertext model, CSP or adversaries can obtain encrypted files, encrypted indexes, and encrypted queries uploaded by DUs and DOs. CSP or adversaries may analyze which of histories are generated by similar keyword sets. If CSP or adversaries can not distinguish indexes/queries through the methods aforementioned, we can say that our scheme is secure in the known ciphertext model.

Theorem 5.1 Our scheme is secure under the known ciphertext model.

Proof Firstly, we need to prove the confidentiality of index/query under the known ciphertext model.

Considering an index $\vec{I_i}$, it is encrypted by Equation(3) and converted by Equation(4). Both of $E_{\mathscr{M}_O}(\vec{I_i})$ and $E_{\mathscr{M}}(\ \vec{I_i})$ can be obtained by CSP or adversaries in known ciphertext model. Therefore the confidentiality of indexes depends on whether CSP or adversaries can obtain effective information from $E_{\mathscr{M}_O}(\vec{I_i})$ and $E_{\mathscr{M}}(\vec{I_i})$.

As shown in Equation (9), the generation of $\{E_{\mathscr{M}_O}(\vec{I_i})$ and $E_{\mathscr{M}}(\ \vec{I_i})$ can both be simplified as the multiplication of a $(2m)$-dimensional vector and a $(2m*2m)$-matrix. Thus the result of encryption is also a $(2m)$-dimensional vector, and there are countless possibilities for it to be decomposed as the product of a vector and a matrix. So as long as CSP or adversaries can not get the secret key, it is difficult for them to deduce the semantic content of the encrypted index.

$$E_{\mathscr{M}}(\vec{I_i})=E_{\mathscr{M}_O}(\vec{I_i}))\times\mathscr{M}'_O=$$

$$(\Gamma\cdot\vec{I_i}+\vec{\varepsilon_i})\times\mathscr{M}_O\times\mathscr{M}'_O=$$

$$(\Gamma\cdot\{\vec{I_i}+\vec{\varepsilon_i})\times\mathscr{M}\tag{9}$$

In addition, we need to proof the indistinguishability of index/query under the known ciphertext model. The indistinguishability of index lies in the encryption algorithm we use and the random number we introduce in the encryption processes. Here we first prove the indistinguishability of index, of which the prerequisite is known ciphertext attack resistance. As shown in Equation (9), the encryption of $\overrightarrow{I_i}$ can be interpreted as the secure KNN encryption of $(\Gamma \cdot \overrightarrow{I_i} + \overrightarrow{\varepsilon_i})$. Thus the known ciphertext attack on $(\Gamma \cdot \overrightarrow{I_i} + \overrightarrow{\varepsilon_i})$ can be resisted, which has already been proved in Reference [14]. Furthermore, based on the random integer vector $\overrightarrow{\varepsilon_i}$ introduced for each $\overrightarrow{I_i}$, we cannot distinguish $(\Gamma \cdot \overrightarrow{I_1} + \overrightarrow{\varepsilon_1})$ and $(\{\Gamma \cdot \overrightarrow{I_2} + \overrightarrow{\varepsilon_2})$ whether are generated from different $\overrightarrow{I_i}$. So, the indistinguishability of index $\overrightarrow{I_i}$ can be guaranteed.

5.2 In known background model

Under the known background model, CSP or adversaries are able to gain a certain number of index and query pairs. They will infer the indexes stored in CSP by matching the known indexes, or analyze the difference of search result by operating multiple queries on the same index. Meanwhile, CSP or adversaries will obtain selected background information such as a certain amount of the keyword and query pairs and try to recover the encrypted indexes through linear analysis. Intuitively, CSP or adversaries cannot distinguish simulator's view from the view he owns. So if a simulator can generate an indistinguishable view, we can say that our scheme is secure under the known background model.

Theorem 5.2 Our scheme is secure under the known background model.

Proof First of all, Yao et al[18] found that if the adversary gets enough ciphertext and corresponding plaintext, he can obtain some information through linear analysing. However, the adversary in our scheme cannot obtain plaintext-ciphertext pair in the known background model. In our scheme, after transforming keywords to vector, it will be extracted with extraction parameters. And these parameters are kept as secret by users. Therefore, even the adversary can obtain some keyword sets, he still cannot get any information through linear analysing under the known background model.

In addition, in order to prove our scheme is secure under the known background model, we denote S as a simulator that can simulate a view V' which is indistinguishable from CSP's view . Then we construct the simulator as follows:

(Ⅰ) we generate F', S select a $\delta_i' \in \{0,1\}^{|\delta_i|}$, $\delta_i \in F, 1 \leqslant i \leqslant \leqslant |F|$ randomly, and outputs $F' = \{\delta_i', 1 \leqslant i \leqslant |F'|\}$.

(Ⅱ) S selects a random invertible matrices $\mathscr{M}' \in \mathbb{Z}_h^{2m*2m}$, setting $SK' = \{\mathscr{M}', \mathscr{M}'^{-1}\}$.

(Ⅲ) S generates query vector Q'_i that number of 1s in q'_i is the same as the number of 1s in Q_i and extends the q'_i by inserting random numbers. Then encrypt it using $\mathscr{M}'^{-1}$ and obtain $\mathrm{Enc}_{\mathscr{M}'^{-1}}(Q')$.

(Ⅳ) S generates $I(F')$ as follow: S first generates a m-bit null vector for $\delta_i' \in F', 1 \leqslant i \leqslant \leqslant |F'|$ as the index, denoted as $I'_{\delta_i'}$. Insert keywords and extend to a 2m-bit vector. Then, S generates $I(F')$ as $\mathrm{Enc}_{\mathscr{M}'}(I_{\delta'_j},$ $1 \leqslant i \leqslant \leqslant |F'|)$. Finally, S outputs the view $V' = (F',$ $\mathrm{Enc}_{\mathscr{M}'}(I(F'))$, $\mathrm{Enc}_{\mathscr{M}'^{-1}}\}(Q'))$.

The construction is correct since the search result on $I(F')$ with the query $\mathrm{Enc}_{\mathscr{M}'^{-1}}(Q')$ is same as the trace which CSP has. Because the hash functions in bloom filter are indistinguishable, the adversary cannot distinguish the output of the linear analysis from a random string. We claim that no probabilistic polynomial-time (P. P. T.) adversary can distinguish the view V' from $V(H)$.

5.3 Privacy preservation in multi-user scenario

In multi-user scenario, different DUs will make queries in the same repository at the same time. These DUs should not be able to know the corresponding queries of other DUs, which can not be guaranteed in some schemes[3,4].

In our scheme, the queries from different DUs for the same repository are encrypted under their own keys M_{O_θ} as Equation (6). So, even if a DU obtains other DUs' $T_{Q_{U_\theta}}$, he/she is still not able to decrypt these queries. After receiving DUs' $T_{Q_{U_\theta}}$, CSP will convert $T_{Q_{U_\theta}}$ to T_Q as Equation (7) which is encrypted under $SK = \mathscr{M}^{-1}$. And nobody has SK to decrypt T_Q except TA.

Therefore, our scheme can achieve privacy preservation in multi-user scenario.

6 Performance evaluation

In this section, we evaluate the performance of our scheme by implementing our system in PYTHON on a Windows 10 server. We use RFC (Request For Comments) database (RFC)[19] as our data set. We choose approximately 1000 files as our experimental data. In the experiments, we set the false positive probability f to 0.01 for the bloom filter. Then, the 20 most frequently occurring words are set as the keywords of files. Meanwhile, the top 10 files will be returned as the final result. Next, we conduct a detailed analysis through our experimental data.

6.1 Comparison

In this subsection, we summarize and compare the existing fuzzy search schemes, of which the results are shown in Table 1. We can see that schemes in References [3,4] use secure KNN to encrypt the index including all the keywords for each file. However, schemes in References [5,6] choose to encrypt letters

Table 1. Comparison with other fuzzy search schemes.

schemes	encrypted object	encryption algorithm	multi-keyword	wildcard query	multi-user set
Wang[3]	index of a file	secure KNN	yes	no	no
Fu[4]	index of a file	secure KNN	yes	no	no
Kim[5]	letter of a keyword	homomorphic encryption	no	yes	no
Yang[6]	substring of a keyword	homomorphic encryption	no	yes	no
Our scheme	index of a file	multi-user set secure KNN	yes	yes	yes

Table 2. Comparison of the Computational Overhead for One Time Search.

schemes	DO		DU		CSP		
	Index generation	index encryption	query generation	query encryption	index conversion	query conversion	search
Wang[3]	rl ∗ H	2m ∗ Dm	rl ∗ H	2nm ∗ Dm	–	–	2n ∗ Dm
Fu[4]	rl ∗ H	2m ∗ Dm	rl ∗ H	2nm ∗ Dm	–	–	2n ∗ Dm
Our scheme	rl ∗ H+L ∗ RT	m ∗ Dm	rl ∗ H+L ∗ RT	m ∗ Dm	m ∗ Dm	nm ∗ Dm	n ∗ Dm

or substrings of a keyword by Homomorphic encryption supporting wildcard query. Encrypting letters or substrings makes schemes[5,6] to support wildcard queries, but this also leads to the scheme only supporting single keyword search. Due to the encryption based on keywords in References [3,4], these schemes can be applied in multi-keyword scenario, but they are unable to support wildcard query. It is noted that all the existing fuzzy search schemes cannot support the multi-user scenario.

The encryption in our scheme is also based on all the keywords in indexes, we introduce a new keyword transformation method to make our scheme able to support both of wildcard query and keyword misspelling. In particular, we apply an improved secure KNN so that our scheme can support multi-user scenario. In summary, only our scheme can support multi-keyword scenario, wildcard query, multi-user scenario simultaneously.

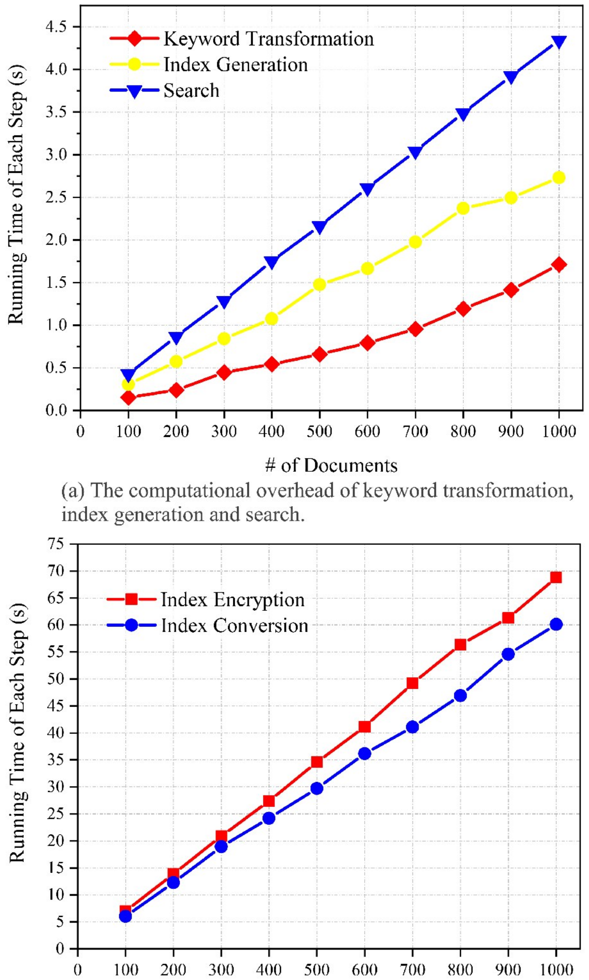
6.2 Efficiency

(Ⅰ) Computational overhead analysis. Here, we compare the computational complexity of our scheme with exiting works[3,4] in the multi-user scenario, which also use the secure KNN to conduct encryption. First of all, we represent the hash operation and rotation operation as H and RT respectively. And we use D_m to denote an m-dimensional dot product operation in the rest parts of this paper for expression simplicity. Specifically, given two m-dimensional vectors $\vec{I}=\{i_1, i_2,\cdots,i_m\}$ and $\vec{Q}=\{q_1,q_2,\cdots,q_m\}$, a D_m operation on them is $\vec{I}\times\vec{Q}=\sum_{j=1}^{m} i_j q_j$. Then the multiplication of a m-dimensional vector and a $(m*m)$-matrix can be expressed as $m*D_m$, and the multiplication of two $(m*m)$-matrixes can be expressed as m^2*D_m.

Now we assume that a DU makes a query for each of the n DOs, where each index/query has r keywords, the average length of all the keywords is L, and the number of hash functions used in bloom filter is l. The computational overhead of index/query generation is $rl*H$ in Wang's and Fu's schemes. But in our scheme, since we introduce the permuterm index, DO/DO needs to conduct extra L times rotation operations during index/query generation. Since the secure KNN we apply is more efficient than that adopted in Wang's and Fu's schemes, the index encryption only has $m*D_m$. But in multi-user scenario, DU in Wang's and Fu's scheme needs to conduct n times encryption on the query for different DOs, therefore the overhead of query encryption is much larger than that in ours scheme, which is $2nm*D_m$. In our scheme, to achieve efficient fuzzy search in multi-user scenario, CSP needs to conduct index and query conversion uponding receiving corresponding ciphertext, of the overhead is $m*D_m$, $nm*D_m$ respectively. But the compared schemes do not have these steps. For the search step, Wang's and Fu's schemes have the same computational overhead, which is $2n*D_m$. And the computational overhead for the search step in our scheme is $n*D_m$. We list the detailed analysis results of the computational overhead in Table 2.

(Ⅱ) Experimental results of computational overhead. Figure 5 shows running time of the keyword transformation, index generation, index encryption, index convertion and search.

Because the index structure in our scheme is a per



(a) The computational overhead of keyword transformation, index generation and search.

(b) The computational overhead of index generation and index encryption.

Figure 5. The computational overhead of our scheme.

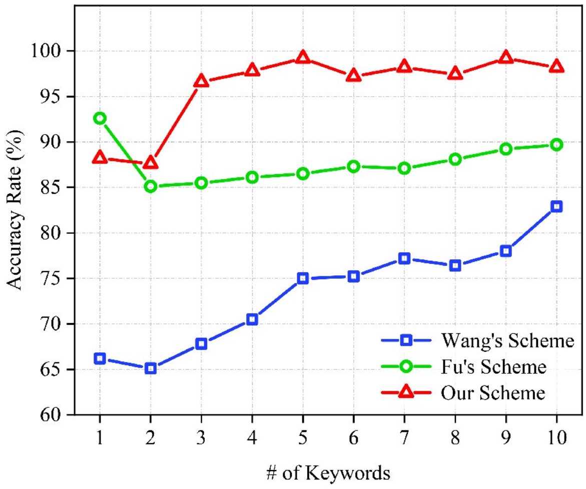


Figure 6. The performance matrices of accuracy with respect to the number of query keywords.

file-based index, each file needs to generate an index and the generation time increased linearly with respect to the number of files. And the running time of index encryption and index convertion are also linear in the number of files. From Figure 6(a), we can see that when the number of files is 1000, the running time of keyword transformation and the index generation are 1.71 seconds and 2.73 seconds, respectively. The key transformation includes word frequency statistics and the generation of substring set. And the running time of search is mainly affected by the number of files. When the number of files is 1000, the search time is approximately 4.3 seconds. Figure 6(b) shows the running time of index encryption and index convertion are also linear in the number of files which are only one-time efforts.

6.3 Result accuracy

In this subsection, We measure result accuracy, which is defined as $\frac{t_p}{t_p+f_p}$. Here t_p represents the true positive, and f_p represents the false positive. The main factors affecting the precision of our scheme are the number of query keywords, the number of fuzzy keywords, and the form of fuzzy keywords. Therefore, our experiment revolves around these three factors.

(Ⅰ) The affection of the number of query keywords. Figure 6 shows the trend of accuracy rate with respect to the number of query keywords when there is one fuzzy keyword in each keyword set.

From Figure 6, it's not hard to see that the search accuracy rates of our scheme are higher than those of the other two schemes when the number of keywords larger than 2, which is always above 95%. The reason for the accuracy improving in our scheme is mainly the use of the new method for index generation.

(Ⅱ) The affection of the number of fuzzy keywords. In order to observe the relationship between the number of fuzzy keywords and the search accuracy, we considered two cases with 5 and 10 query keywords respectively, of which the results are shown in Figure 7 and Figure 8. In Figure 7, due to the number of query keywords is small, the overall accuracy rate shows a downward trend. But our scheme's accuracy rate decreases much more slowly than Wang's and Fu's schemes. And the accuracy rate of our scheme is still acceptable when other two schemes could not return the correct result. In the case with 10 query keywords, the accuracy rate of our scheme appears a turning point when the number of fuzzy keywords reaches 7, which is shown in Figure 8. Then, the accuracy rate of our scheme is gradually reduced and reaches the minimum value of 70% when all keywords are fuzzy keywords. While the accuracy rate of other two schemes is decreased from 100% to 10% sharply as the number of fuzzy keywords is increased from 6 to 10.

(Ⅲ) The affection of the type of fuzzy keywords: We first compare our scheme with schemes[3-6] in terms of the capabilities of dealing with all possible types of

Table 3. Different Types of Fuzzy Keyword.

query keyword	index keyword	Our scheme	Wang[3]	Fu[4]	Kim[5]	Yang[6]
im * vement	improvement	yes	no	no	yes	yes
improv*		yes	no	no	yes	yes
*provement		yes	no	no	yes	yes
imprvoement		yes	yes	yes	no	no
imprdvement		yes	yes	yes	no	no
imprment		no	yes	yes	no	no
improvnmement		no	yes	yes	no	no
thing	night	no	yes	no	no	no

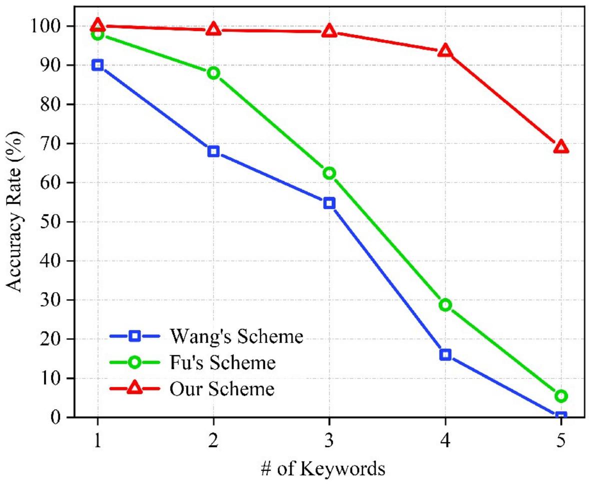


Figure 7. The accuracy rates with respect to the number of fuzzy keywords in the case of 5 query keywords.

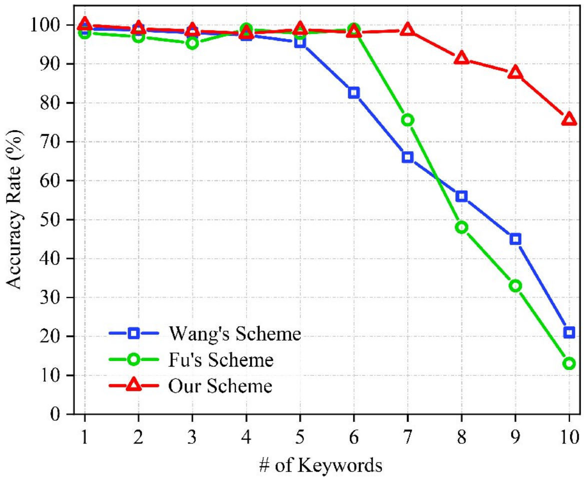


Figure 8. The accuracy rates with respect to the number of fuzzy keywords in the case of 10 query keywords.

fuzzy keywords, which is illustrated by Table 3. “yes” means that the index keyword and the query keyword can be matched, and “no” means that they can not be matched. We can see that wildcard query keyword like “im * vement, improv * , * provement” can be solved in our scheme, Kim’s scheme, and Yang’s scheme. General spelling error like “imprvoement, imprdvement” can be solved in our scheme, Wang’s scheme, and Fu’s scheme. And keywords composed of the same letters in different order like “thing, night” can be distinguished in our scheme and Fu’s scheme. In summary, other schemes are only able to deal with a subset of keywords while our scheme is able to deal with all types of fuzzy keywords.

Then, we measure the accuracy rates when keywords are displayed in different forms, including replacing letters (RL), wildcard query (WQ), and reversing the order of two letters (ROTL). The results are shown in Figure 9. In this part, we set at least 4 fuzzy keywords in a query. In the case of ROTL, we randomly select two keywords and exchange the two letters in each keyword, e. g. , from “program” to “prgoram”. The accuracy rates grow from 75% to 90% as the number of query keywords is increased from 4 to 6, and then it stays at 95%. And in order to simulate RL, we randomly choose several letters and replace them with other letters for each keyword. The general trend of RL is the same as that of ROTL. There is a growth period as the number of keywords is increased from 4 to 6, and a stable period from 6 to 10. Meanwhile, RL performs better than ROTL when the fuzzy keywords account for a large proportion. When the number of accurate keywords are more than that of fuzzy keywords, the accuracy of ROTL is higher. As for the case of WQ, we randomly replace several letters with wildcard character “ * ”. Different from RL and ROTL, the accuracy rate of WQ always stay above 90% as the number of keywords is increased from 4 to 10. Obviously, the accuracy rate of WQ is not affected by the proportion of fuzzy keywords, and it always maintains high accuracy.

The reason for such results is that keyword

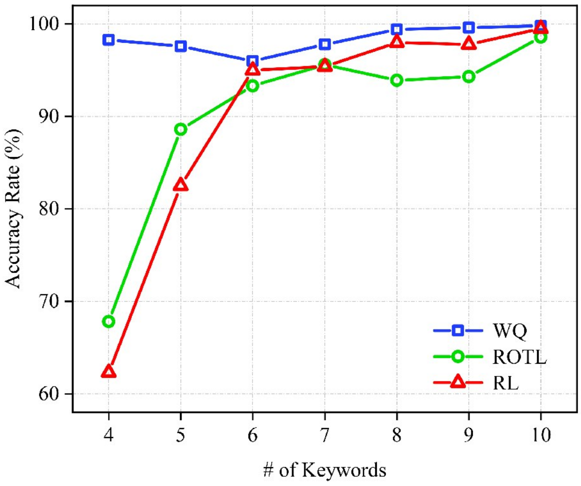


Figure 9. The accuracy rates with respect to the types of fuzzy keywords.

transformation in our scheme is the permuterm index which is aimed at wildcard query. From these three kinds of fuzzy search, we note that the correct results can be returned when the proportion of fuzzy keywords is more than 50%. And our scheme is the most suitable for the wildcard query.

According to the above experiments, we can draw a conclusion that our scheme is able to maintain high precision in various cases.

7 Relate work

7.1 Searchable encryption

The first construction of searchable encryption[20] was proposed by Song et al. in 2000. On this basis, a lot of work is committed to designing an effective searchable encryption scheme. Cash et al[21] proposed a scalable symmetric searchable encryption (SSE) scheme to support boolean query. To improve the search experience, the multi-keyword rank searchable encryption (MRSE) mechanism is proposed[22-25] where CSP can return the top-k results according the similarity of keyword sets. At present, the research direction of searchable encryption can be roughly divided into the functional improvement of public key encryption with keyword search (PEKS) and the dynamic security of symmetric searchable encryption (SSE)[26-29]. Existing schemes of PEKS mainly focus on multi-keyword search[7,30] and authorized keyword search[31-35].

7.2 Fuzzy search over encrypted data

Noteworthy, the above schemes only support accurate keyword searches. Thus many attentions have been drawn to fuzzy keyword search over encrypted data. Concretely, the first work[36] focusing on fuzzy search over encrypted data exploited edit distance to quantify keywords similarity between the predefined extend keyword set and the fuzzy keyword. But it only supports the single keyword fuzzy search and needs to predefine extend keyword set. Reference [3] proposed a multi-keyword fuzzy searchable encryption scheme without a predefined fuzzy set. In their work, a keyword is first transformed into a substring set containing the bi-grams of the keyword and evaluate keywords similarity according to the Euclidean distance. For example, the substring set of “fuzzy” is {“fu”, “uz”, “zz”, “zy”}. The problem of this approach is that it’s only able to solve the mistakes with only one letter misspelled. To improve the tolerance of misspelling, Fu et al[4] designed a new method of keyword transformation. Differently, the keyword “fuzzy” is represented as {“f1”, “u1”, “z1”, “z2”, “y1”}. And schemes in References [5, 6] introduce wildcard to represent any character in order to tolerate more complicated keyword misspelling. In their work, they choose to split up the keyword and encrypt the substrings of each keyword by homomorphic encryption. Obviously, the computational overhead of these two schemes is extremely high and their work can only solve wildcard query. Therefore, how to improve the fault tolerance and accuracy while bringing less computational overhead is the main problem in fuzzy search over encrypted data.

7.3 Searchable encryption considering multi-user set

In cloud storage scenario, multiple users upload and download files, the access permission between data owners and data users is a many-to-many relationship. For the calculation between query and index, data users must encrypt their queries under the corresponding key of the data owner. The data owner has to share his/her secret key with data users to support keyword searching, which may lead to privacy disclosure. Meanwhile, it is necessary for data users to encrypt the same query index multiple times for different files from different data owners, which greatly increases the overhead of communication and computation.

Regarding the above challenges in the multi-user setting, many efforts have been made on the searchable encryption by applying a new encryption algorithm[7-9]. Yang et al applied a new homomorphic encryption scheme in a traditional keyword searchable encryption scheme[7], where the calculation result of data encrypted by different keys can be decrypted by the sum of two keys. Considering image retrieval in the multi-user scenario, the scheme in Reference [8] applies an improved encryption algorithm to encrypt images. As for Reference [9], Cheng et al solved the problem of multi-key searchable encryption in database. Based on homomorphic encryption, they introduced two cloud to conduct collaborate computing, ensuring the computability and privacy of data. However, these schemes cannot efficiently or effectively be applied to fuzzy search.

8 Conclusion

In this paper, we proposed a multi-user multi-keyword fuzzy search scheme over encrypted cloud storage. We utilized permuterm index to design a new keyword transformation method, which enables our scheme to achieve wildcard query, making the scheme able to deal with more varieties of misspellings in keywords. By replacing the LSH used in the bloom filter to standard hash function, our scheme improves the accuracy rate when there are multiple fuzzy keywords. Moreover, our scheme adopts a new efficient key conversion protocol where encrypted indexes and queries are re-encrypted by CSP. In such way, our scheme can achieve efficient fuzzy search in the multi-user scenario. Through comprehensive security analysis and the experiments on real data set, we demonstrated that our scheme is efficient, effective, and feasible in the area of searchable encryption.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (61972371) and Youth Innovation Promotion Association of Chinese Academy of Sciences (CAS) (Y202093).

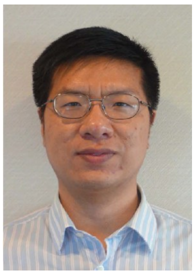
Conflict of interest

The authors declare no conflict of interest.

Author information

LING Huaze received her bachelor's degree from the School of Computer Science and Technology, Hefei University of Technology in July, 2018. She is currently a graduate student in Information Security from the School of Cyber Security, USTC. Her research interests include Network security and Cryptography.

XUE Kaiping (M'09-SM'15) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and the PhD degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering, University of Florida. He is currently a Professor with the School of Cyber Security and the Department of EEIS, USTC. His research interests include future Internet architecture, transmission optimization, distributed networks and network security. He is a fellow of the IET and a senior member of the IEEE. He serves on the Editorial Board of several journals, including the IEEE Transactions on Wireless Communications and the Ieee Transactions on Network and Service Management.



David S. L. WEI (SM'07) received his PhD degree in Computer and Information Science from the University of Pennsylvania in 1991. From May 1993 to August 1997 he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan. He has authored and co-authored more than 100 technical papers in various archival journals and conference proceedings. He is currently a Professor of Computer and Information Science Department at Fordham University. His research interests include cloud computing, big data, IoT, and cognitive radio networks. He was a guest editor or a lead guest editor for several special issues in the IEEE Journal on Selected Areas in Communications, the Ieee Transactions on Cloud Computing and the IEEE Transactions on Big Data. He Also Served as an Associate Editor of Ieee Transactions on Cloud Computing, 2014 - 2018, and an Associate Editor of Journal of Circuits, Systems and Computers, 2013-2018.

LI Ruidong (SM'07) is an associate professor in College of Science and Engineering, Kanazawa University, Japan. Before joining Kanazawa University, he was an a senior researcher with the Network System Research Institute, National Institute of Information and Communications Technology (NICT). He received a bachelor in engineering from Zhejiang University, China, in 2001. He received a doctorate of engineering from the University of Tsukuba in 2008. He is the founder and chair for the IEEE SIG on big data intelligent networking and IEEE SIG on intelligent Internet edge. He also serves as the chairs for conferences and workshops and organized the special issues for the leading magazines and journals. His current research interests include future networks, big data networking, blockchain, information-centric network, internet of things, network security, wireless networks, and quantum Internet. He is a senior member of the IEEE and a member of IEICEasd.

References

[1] Wang B, Yu S, Lou W, et al. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. Proceedings of the International Conference on Computer Communications (INFOCOM). IEEE, 2014: 2112-2120.

[2] Fu Z, Wu X, Guan C, et al. Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Transactions on Information

Forensics and Security，2016，11（12）：2706-2716.

[3] Kim M，Lee H T，Ling S，et al. Private compound wildcard queries using fully homomorphic encryption. IEEE Transactions on Dependable and Secure Computing，2019，16（5）：743-756.

[4] Yang Y，Liu X，Deng R H，et al. Flexible wildcard searchable encryption system. IEEE Transactions on Services Computing，2020，13（3）：464-477.

[5] Yang Y，Liu X，Deng R. Multi-user multi-keyword rank search over encrypted data in arbitrary language. IEEE Transactions on Dependable and Secure Computing，2017，17（2）：320-334.

[6] Wang X，Ma J，Liu X，et al. Search in my way：Practical outsourced image retrieval framework supporting unshared key. Proceedings of the International Conference on Computer Communications（INFOCOM）. IEEE，2019：2485-2493.

[7] Cheng K，Shen Y，Wang Y，et al. Strongly secure and efficient range queries in cloud databases under multiple keys. Proceedings of the International Conference on Computer Communications（INFOCOM）. IEEE，2019：2494-2502.

[8] Curtmola R，Garay J，Kamara S，et al. Searchable symmetric encryption：Improved definitions and efficient constructions. Journal of Computer Security，2011，19（5）：895-934.

[9] Bloom B H. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM，1970，13（7）：422-426.

[10] Mitzenmacher M. Compressed bloom filters. IEEE/ACM Transactions on Networking，2002，10（5）：604-612.

[11] Broder A Z，Mitzenmacher M. Network applications of bloom filters：A survey. Internet Mathematics，2004，1（4）：485-509.

[12] Wong W K，Cheung D W-L，Kao B，et al. Secure kNN computation on encrypted databases. Proceedings of the ACM SIGMOD International Conference on Management of Data（SIGMOD）. Rhode Island，USA：ACM，2009：139-152.

[13] Blaze M，Bleumer G，Strauss M. Divertible protocols and atomic proxy cryptography. International Conference on the Theory and Application of Cryptographic Techniques. Espoo，Finland：Springer，1998：127-144.

[14] Ateniese G，Fu K，Green M. Improved proxy reencryption schemes with applications to secure distributed storage. ACM Transactions on Information and System Security，2006，9（1）：1-30.

[15] Yuan J，Tian Y. Practical privacy-preserving MapReduce based *k*-means clustering over large-scale dataset. IEEE Transactions on Cloud Computing，2017，7（2）：568-579.

[16] Yao B，Li F F，Xiao X K. Secure nearest neighbor revisited. International Conference on Data Engineering. Brisbane，Australia：IEEE，2013：733-744.

[17] RFC Index. https://www.rfc-editor.org/rfc-index.html/.

[18] Song D X，. Wagner D，Perrig A. Practical techniques for searches on encrypted data. Proceedings of IEEE Symposium on Security and Privacy（S&P）. Berkeley，USA：IEEE，2000：44-55.

[19] Cash D，Jarecki S，Jutla C. et al. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. Berlin Heidelberg：Springer，2013.

[20] Cao N，Wang C，Li M，et al. Privacy preserving multi-keyword ranked search over encrypted cloud data. IEEE Transactions on Parallel and Distributed Systems，2014，25（1）：222-233.

[21] Fu Z，Sun X，Linge N，et al. Achieving effective cloud search services：Multi-keyword ranked search over encrypted cloud data supporting synonym query. IEEE Transactions on Consumer Electronics，2014，60（1）：164-172.

[22] Sun W，Wang B，Cao N，et al. Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Transactions on Parallel Distributed Systems，2014，25（11）：3025-3035.

[23] Yu J，Lu P，Zhu Y，et al. Toward secure multikeyword top-*k* retrieval over encrypted cloud data. IEEE Transactions on Dependable Secure Computing，2013，10（4）：239-250.

[24] Sun S F，Yuan X，Liu J K，et al. Practical backward-secure searchable encryption from symmetric puncturable encryption. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security（CCS）. Toronto，Canada：ACM，2018：763-780.

[25] Chamani J G，Papadopoulos D，Papamanthou C，et al. New constructions for forward and backward private symmetric searchable encryption. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security（CCS）. Toronto，Canada：ACM，2018：1038-1055.

[26] Bost R. ∑οφος：Forward secure searchable encryption. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security（CCS）. Vienna Austria：ACM，2016：1143-1154.

[27] Song X，Dong C，Yuan D，et al. Forward private searchable symmetric encryption with optimized I/O efficiency. IEEE Transactions on Dependable and Secure Computing，2020，17（5）：912-927.

[28] Ding X，Liu P，Jin H. Privacy-preserving multi-keyword top-*k* similarity search over encrypted data. IEEE Transactions on Dependable and Secure Computing，2017，16（2）：344-357.

[29] Sun W，Yu S，Lou W，et al. Protecting your right：Attribute-based keyword search with fine-grained owner enforced search authorization in the cloud. IEEE Transactions on Parallel and Distributed Systems，2016，27（4）：1187-1198.

[30] Xu L，Chen X，Zhang F，et al. ASBKS：Towards attribute set based keyword search over encrypted personal health records. IEEE Transactions on Dependable and Secure Computing，2020，https://doi.org/10.1109/TDSC.2020.2970928.

[31] Yang Y，Liu X，Deng R H，et al. Lightweight sharable and traceable secure mobile health system. IEEE Transactions on Dependable and Secure Computing，2020，17（1）：78-91.

[32] Liu X，Yang G，Mu Y，et al. Multi-user verifiable searchable symmetric encryption for cloud storage. IEEE Transactions on Dependable and Secure Computing，2020，17（6）：1322-1332.

[33] Zhang K，Wen M，Lu R，aet al. Multi-client sublinear

boolean keyword searching for encrypted cloud storage with owner-enforced authorization. IEEE Transactions on Dependable and Secure Computing, 2020, PP(99): 1-1.

[34] Kermanshahi S K, Liu J K, Steinfeld R, et al. Multi-client cloud-based symmetric searchable encryption. IEEE Transactions on Dependable and Secure Computing. 2021, 18(5): 2419-2437.

[35] Li J, Wang Q, Wang C, et al. Fuzzy keyword search over encrypted data in cloud computing. Proceeding of the 2010 International Conference on Computer Communications (INFOCOM). San Diego, USA: IEEE, 2010: 1-5.

多用户场景支持多关键词模糊搜索的可搜索加密方案

凌华泽[1], 薛开平[1*], David S. L. WEI[2], 李瑞东[3]

1. 中国科学技术大学网络空间安全学院,安徽合肥 230027
2. 美国福特汉姆大学计算机与信息科学系,美国纽约,10458
3. 日本金泽大学科学与工程学院,石川金泽,920-1192
*通讯作者. E-mail:kpxue@ustc.edu.cn

摘要: 随着云计算平台的普及与推广,越来越多的企业和个人选择将数据外包到云以降低本地的维护成本,因此用于解决加密数据上关键字搜索问题的可搜索加密技术(searchable encryption,SE)变得越来越重要. 模糊搜索概念的引入主要是为了解决查询关键词出现错误的情况. 然而,现有的支持模糊搜索的可搜索加密方案不仅在多用户场景中具有很高的计算和通信开销,而且不能在保证有效准确率的前提下解决各种关键词错误类型. 为此提出了一种多用户场景下支持多关键字模糊搜索的可搜索加密方案. 具体来说,我们引入轮排索引来支持多关键字通配符搜索,可以以更高的正确率支持更多类型的模糊关键词. 此外,通过让云服务器对索引信息进行重加密,本方案支持多用户场景非密钥共享的模糊搜索,有效降低了用户的计算开销并提高了隐私保护水平. 实验结果表明,与现有方案相比,该方案不仅具有较高的准确率,而且能够以可接受的计算开销支持多种拼写错误的关键字搜索.

关键词: 加密云存储;代理重加密;隐私保护;可搜索加密