IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 8, 2018, accepted August 10, 2018, date of publication August 17, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2866031

Enabling Efficient Verifiable Fuzzy Keyword Search Over Encrypted Data in Cloud Computing

XINRUI GE[1], JIA YU[1,2,3], CHENGYU HU[4], HANLIN ZHANG[1], AND RONG HAO[1]

[1]College of Computer Science and Technology, Qingdao University, Qingdao 266071, China
[2]State Key Laboratory of Cryptology, Beijing 100878, China
[3]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[4]School of Software, Shandong University, Jinan 250101, China

Corresponding author: Jia Yu (qduyujia@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572267, Grant 61272425, Grant 61402245, and Grant 61602275, in part by the National Development Foundation of Cryptography under Grant MMJJ20170118 and Grant MMJJ20170126, in part by the Open Project of Co-Innovation Center for Information Supply and Assurance Technology, Anhui University, under Grant ADXXBZ201702, in part by the Open Project of the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences under Grant 2017-MS-21 and Grant 2016-MS-23, in part by the Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, NJUPT, under Grant BDSIP1806, and in part by the Natural Science Foundation of Shandong Province under Grant ZR2015FM020 and Grant ZR2016FQ02.

ABSTRACT Searchable encryption can support data user to selectively retrieve the cipher documents over encrypted cloud data by keyword-based search. Most of the existing searchable encryption schemes only focus on the exact keyword search. When data user makes spelling errors, these schemes fail to return the result of interest. In searchable encryption, the cloud server might return the invalid result to data user for saving the computation cost or other reasons. Therefore, these exact keyword search schemes find little practical significance in real-world applications. In order to address these issues, we propose a novel verifiable fuzzy keyword search scheme over encrypted cloud data. For the purpose of introducing this scheme, we first propose a verifiable exact keyword search scheme and then extend this scheme to the fuzzy keyword search scheme. In the fuzzy keyword search scheme, we employ the linked list as our secure index to achieve the efficient storage. We construct a linked list with three nodes for each exact keyword and generate a fuzzy keyword set for it. To reduce the computation cost and the storage space, we generate one index vector for each fuzzy keyword set, rather than each fuzzy keyword. To resist malicious behaviors of the cloud server, we generate an authentication label for each fuzzy keyword to verify the authenticity of the returned ciphertexts. Through security analysis and experiment evaluation, we show that our proposed schemes are secure and efficient.

INDEX TERMS Cloud computing, searchable encryption, verifiable exact keyword search, verifiable fuzzy keyword search.

I. INTRODUCTION

Nowadays, cloud computing is playing an increasingly important role in daily lives. As a promising computing model, it provides us with scalable computing resources, on-demand high quality services and ubiquitous network access. Owing to its convenience and flexibility, more and more data owners prefer to store their data into the cloud server for reducing the storage space and saving the management overhead at local machines. Because the cloud server is in different trusted domains with the data owner, the data is out of the owner's physical control. The data owner needs to be able to check whether the data is correctly stored on cloud [1]–[3]. In addition, the cloud data may be known by the malicious cloud server or undelegated users(e.g.hackers), which results in the disclosure of the owner's private data. For instance, in 2016, the second largest social networking site in the world, Myspace, was hacked, which leads to the leak of users' account information. Therefore, it is of imminent importance to protect the privacy of the cloud data.

To secure the data on the cloud server, the data owner usually encrypts the data before uploading them to the cloud server. However, it is intractable for the data user to search over the encrypted cloud data by adopting the traditional search scheme. One simple solution to this issue is that the data user downloads all of the cipher data from the cloud server and then decrypts them locally. However, it will incur

2169-3536 © 2018 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

a huge amount of network bandwidth, storage space and computation overhead, which is infeasible actually. Another possible way is that the user provides the decryption key and the queried keyword to the cloud server. The cloud server decrypts the cipher data and performs search operation over the plaintext. Obviously, this way will expose the data to the cloud server, which is improbable. In order to address this issue, many searchable encryption(SE) schemes have been proposed [4]–[6], [8]–[14]. SE allows the data user to selectively retrieve cipher documents stored on the cloud server by keyword-based search.

Up to now, most of these schemes assume that the cloud server is honest-but-curious. In other words, the cloud server follows the pre-defined protocol to perform search operation over encrypted cloud data, but prefers to know extra information from the search trapdoor and the secure index. However, in real scenario, the cloud server is not always honest. It may return invalid search result for saving the computation cost. Thus, it is necessary for the data user to verify the search result. Kurosawa and Ohtaki *et al.* [15] proposed a UC-secure verifiable searchable symmetric encryption (SSE) scheme. This scheme first formulates the security of verifiable SSE schemes against active adversaries by introducing the notion of privacy and reliability. It solves the problem that the same storage location is padded with different values in scheme [5]. The computation cost of verification is linear with the number of documents. In addition, this scheme can only support the exact keyword search, while the data user often makes mistakes when inputting keywords. In this case, the exact keyword search might not return the documents of interest. Therefore, it is necessary to design a scheme supporting fuzzy keyword search. Li *et al.* [16] firstly proposed the fuzzy keyword search scheme over encrypted cloud data. In this scheme, edit distance is used to quantify keywords similarity. This scheme develops a novel technique, i.e., an wildcard-based technique, for the construction of fuzzy keyword sets. This technique eliminates the need for enumerating all the fuzzy keywords. The resulted size of the fuzzy keyword sets is significantly reduced. However, this scheme does not consider the verification for search result. Wang *et al.* [17] proposed the first verifiable fuzzy keyword search scheme, which not only enables fuzzy keyword search over encrypted data, but also maintains keyword privacy and the verifiability of the search result. Zhu *et al.* [18] introduced a verifiable dynamic fuzzy keyword search scheme, which is UC-secure against the malicious adversary. The verification process incurs enormous time cost since it adopts RSA accumulator based on the public key system to authenticate the search result. Therefore, how to realize the efficient verifiable fuzzy keyword search is worth studying.

Contributions:

1) Before introducing the main work, we propose a verifiable exact keyword search (VEKS) scheme, which achieves search operation in one communication round. To detect the malicious behavior of the cloud server, we generate an authentication label for each keyword. After the cloud server returns the search result, the data user can verify the validity of search result based on the authentication label. With this method, we can detect whether the documents returned from the cloud server have been modified or deleted.
2) Based on the proposed VEKS scheme, we propose a verifiable fuzzy keyword search (VFKS) scheme. In the VFKS scheme, in order to efficiently build the index, we adopt the linked lists as the index structure. We store a fuzzy keyword set rather than one single fuzzy keyword into a node, and only generate one index vector for each fuzzy keyword set instead of each single fuzzy keyword. This method can greatly reduce the storage space. In addition, we do not need to construct the fuzzy keyword set for the queried keyword in our design, which reduces the computation cost and improves the search efficiency.
3) Through the security analysis and experimental evaluation on a large real-world dataset, we show that our proposed schemes are secure and efficient.

Organization: The rest of this paper is organized as follows. In Section II, we introduce the related work. Some preliminaries are given in Section III. Section IV proposes the two schemes, the verifiable exact keyword search scheme and the verifiable fuzzy keyword search scheme. Section V gives the security analysis. Section VI gives the performance analysis and the experiment results. We give the further discussion in Section VII. Section VIII concludes the paper.

II. RELATED WORK

A. EXACT KEYWORD SSE

Song *et al.* [4] firstly proposed the searchable symmetric encryption scheme. In this scheme, a special two-layered encryption structure is constructed to encrypt each keyword. The cloud server must search all documents with a sequential scan. So the search time cost is linear with the total size of the document collection. Curtmola *et al.* [5] proposed two efficient searchable symmetric encryption schemes. These schemes support multi-user to submit the search request, and realize sublinear search, that is, the search cost is proportional to the number of documents containing the queried keyword. Cao *et al.* [13] firstly proposed a privacy-preserving multi-keyword ranked search scheme over encrypted cloud data. This scheme adopts the similarity measure of ''Coordinate matching''to capture the relevance of documents with the queried keyword, and uses ''inner product similarity'' to quantitatively evaluate such similarity measure. Xia *et al.* [6] proposed a secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. This scheme adopts a tree-based index, which is combined with the vector model and the TF $\times$ IDF model. Fu *et al.* [7] built a user interest model for individual user by analyzing the users search history to enable user to retrieve relevant documents from the cloud server based on his own interest. In addition, some other SSE schemes have also been proposed, such as conjunctive

keyword query scheme [8], ranked search scheme [11], [12], ranked multi-keyword search scheme for multiple data owners [12], [20], central keyword-based semantic extension search scheme [19], and keyword search schemes supporting deduplication [9], [10].

B. FUZZY KEYWORD SSE

When the data user makes spelling errors, the traditional exact keyword search schemes cannot return the documents of interest. It greatly affects the system usability. To address this problem, Li *et al.* [16] firstly proposed a fuzzy keyword search scheme over encrypted cloud data using the edit distance to measure the similarity of two keywords. This scheme adopts the wildcard technology to construct the fuzzy keyword sets. Kuzu *et al.* [21] proposed a similarity searchable symmetric encryption scheme, which makes use of minhash based on Jaccard distance to support fault tolerant keyword search. Wang *et al.* [22] proposed the first multi-keyword fuzzy keyword search scheme based on the Bloom filter and the LSH function, in which the keyword is transformed into a bi-gram set. Wang *et al.* [23] proposed a range fuzzy keyword search scheme by using a score table structure to rank the documents. Fu *et al.* [24] proposed a multi-keyword fuzzy ranked search scheme based on scheme [22]. This scheme designs a new method of keyword transformation based on the unigram, and considers the keyword weight when selecting an adequate matching file set.

C. VERIFIABLE SSE

The cloud server may return the incorrect search result to the data user for saving the computation cost or other reasons. To address this problem, it is necessary for the data user to verify the search result. Chai and Gong [25] firstly proposed a verifiable keyword search scheme over encrypted cloud data. In this scheme, the cloud server needs to prove that the returned result is correct, which is named as the ''verifiable searchability''. Wang *et al.* [26] proposed a verifiable keyword search scheme by using the hash chain to verify the search result. Kurosawa and Ohtaki [15] proposed the first UC-secure verifiable SSE scheme based on scheme [5]. This scheme can verify whether the search result is modified or deleted. The computation cost of verification is linear with the number of documents. Sun *et al.* [27] proposed an efficient verifiable conjunctive keyword search scheme, in which the authors exploit a bilinear-map accumulator tree as the authenticated data structure. Wang textitet al. [28] proposed a verifiable keyword search scheme based on the inverted bloom filter without the process of pre-counting. Furthermore, it realizes the multi-user setting by incorporating multi-party searchable encryption (MPSE), which can resist the collusion attack between the cloud server and the malicious data user. Jiang *et al.* [29] proposed a verifiable multi-keyword ranked search scheme over encrypted cloud data, in which it constructs a special data structure named QSet to achieve efficient keyword search. These verifiable keyword search schemes can work only when the data user's input is completely correct. However, the data user may make spelling errors in real scenario. To address this problem, some verifiable fuzzy keyword search schemes have been proposed. Wang *et al.* [17] proposed the first verifiable fuzzy keyword search scheme, which not only enables fuzzy keyword search over encrypted data, but also maintains keyword privacy and the verifiability of the search result. Zhu *et al.* [18] introduced a verifiable dynamic fuzzy keyword search scheme, which is UC-secure against a malicious adversary. The verification process incurs enormous time cost since it adopts RSA accumulator based on the public key system to authenticate the search result. In addition, verifiable multiple users search scheme [30], verifiable ranked keyword search scheme [31], verifiable keyword search scheme in multi-owner settings [32], verifiable keyword-based semantic search scheme [33] and verifiable dynamic keyword search scheme [34] have also been proposed.

III. PROBLEM FORMULATION

A. SYSTEM MODEL

As shown in Fig.1, the system model involves three different entities: the data owner, the data user and the cloud server. The data owner encrypts the plaintexts $\mathcal{F} = (F_1, F_2, \ldots, F_N)$ into ciphertexts $\mathcal{C} = (C_1, C_2, \ldots, C_N)$. In order to realize the efficient search over the ciphertexts, the data owner constructs a secure index $\mathcal{I}$ for all different keywords extracted from plaintexts $\mathcal{F}$. To verify the correctness of the returned result, the data owner computes an authentication label (*tag*) for each keyword and stores all of the authentication labels into a table named as *Tag* table. Finally, the data owner uploads the encrypted files $\mathcal{C}$, the secure index $\mathcal{I}$ and the *Tag* table to the cloud server. When the data user wants to search some documents containing the queried keyword w, he computes the trapdoor of w, and sends the trapdoor to the cloud server. Upon receiving the trapdoor from the data user, the cloud server performs search operation over the ciphertexts, and returns the related encrypted files and the authentication label to the data user. After the data user receives the search result from the cloud server, he will verify whether it is valid. If it is valid, accepts it; otherwise, rejects it.

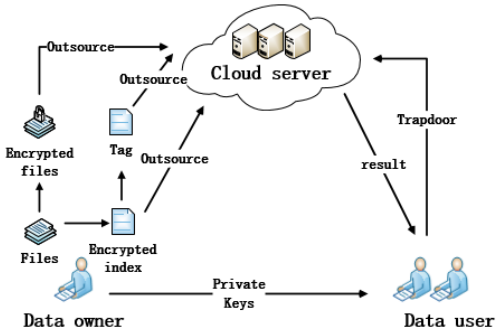


FIGURE 1. The framework of the system model.

B. THREAT MODEL

In the threat model, the data owner and the data user are assumed to be always trusted. That is, the data owner honestly encrypts documents, builds secure index, and computes authentication labels. The data user honestly generates the trapdoor for the queried keyword. The cloud server is regarded as an untrusted entity. It may try to learn additional valuable information from the encrypted files, the secure index and the search trapdoor. Moreover, it may return invalid search result to the data user for saving computation cost.

Besides the encrypted documents, the secure index and the search trapdoor, the cloud server can know and record each search result. We name this type of knowledge as *known ciphertext*. The cloud server also knows some background information e.g. the relationship of two search trapdoors, that is, which documents commonly contain two queried keywords. This type of knowledge is named as *known background*.

C. DESIGN GOALS

Our scheme should satisfy the following requirements:

- *Keyword search.* The scheme should be able to retrieve all documents containing the queried keyword.
- *Efficiency.* The scheme should efficiently return the search result and incur little overhead on verifying the result.
- *Privacy preserving.* The scheme cannot reveal any data information to the cloud server, such as the plaintext of keywords and documents beyond the limited leakage. Usually, we consider the limited leakage from the following three aspects.
  1) *Documents privacy.* The contents of the documents cannot be leaked but the number of documents.
  2) *Index privacy.* The cloud server cannot find the storage relationship between keywords and documents from the secure index.
  3) *Trapdoor privacy.* It should guarantee the trapdoor cannot reveal any information about the queried keyword.
- *Search results verifiability.* The scheme should be able to verify whether the search result is true to prevent the cloud server from returning the incorrect search result to the data user.

D. NOTATIONS AND PRELIMINARIES

We define some notations in Table.1 to help readers read our paper more easily.

Cryptographic tools are very important for designing cryptographic schemes [35], [36]. Some crytograhpic tools, concepts and structures used in this paper are introduced as follows.

1) SYMMETRIC ENCRYPTION

A symmetric encryption scheme includes three polynomial-time algorithms SKE $=$ (Gen, Enc, Dec). Algorithm Gen is used to generate the secret key sk by inputting a secure parameter k. Algorithm Enc is used to encrypt the plaintext F into the ciphertext C with sk. Algorithm Dec is used to decrypt the ciphertext C into the plaintext F with sk. A symmetric encryption scheme should be secure against chosen-plaintext attack. We select the AES encryption algorithm that satisfies above requirement to encrypt the documents.

TABLE 1. Some frequently used symbols and descriptions.

Symbols	Descriptions
$\mathcal{W}$	the different keywords collection, $\mathcal{W} = (w_1, w_2, ..., w_n)$
w_i	the i-th keyword in $\mathcal{W}$
n	the number of keywords
$\mathcal{F}$	the document collection, $\mathcal{F} = (F_1, F_2, ..., F_N)$
F_j	the j-th document in $\mathcal{F}$
N	the number of documents
$\mathcal{C}$	the encrypted document collection, $\mathcal{C} = (C_1, C_2, ..., C_N)$
C_j	the ciphertext of F_j
K	the private key set $K = \{sk, k_0, k_1, k_2\}$
T_w	the search trapdoor of keyword w
$\mathcal{I}$	the secure index
$F(w)$	the document collection containing w
$C(w)$	the encrypted documents collection containing w
$v(w)$	the index vector of w
$Ev(w)$	the encrypted index vector of w
tag_w	the authentication label of w
$S_{w_i,d}$	the fuzzy keywords set of w_i with edit distance d
d	the edit distance
$w_{i,j}$	the j-th fuzzy keyword of w_i

2) PSEUDO-RANDOM FUNCTION AND MESSAGE AUTHENTICATION FUNCTION

Pseudo-random functions(PRF) and pseudo-random permutation functions(PRP) are polynomial-time computable functions, which cannot be distinguished from random functions by any probabilistic polynomial-time adversary. We make use of PRF f to blind the index vectors and PRP π to confuse the location of keywords. The parameters of above two functions are as follows:

$$\pi : \{0,1\}^k \times \{0,1\}^l \rightarrow \{0,1\}^l,$$
$$f : \{0,1\}^k \times \{0,1\}^l \rightarrow \{0,1\}^N,$$

where l is the most length of keywords.

Let MAC: $\{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^N$ be an authentication label generation function, which has irreversibility and message unforgeability against chosen message attacks [37]. We use it to generate an authentication label for message and verify the correctness of search result returned from the cloud server. In this paper, we write $tag = \mathrm{MAC}(m)$ instead of $tag = \mathrm{MAC}(k_0, m)$, where k_0 is a key and m is a message.

3) INVERTED INDEX

In our paper, we build the index table based on the inverted index [5], as shown in Table.2. Firstly, the data owner scans the whole documents to extract all of the distinct keywords. Then he constructs $F(w_i)$ for each keyword w_i to denote the set of documents containing the keyword $w_i (1 \leq i \leq n)$.

TABLE 2. Inverted index.

W \ F	F_1	F_2	F_3	$\cdots$	F_N
w_1	1	0	1	$\cdots$	0
w_2	0	1	0	$\cdots$	1
w_3	1	0	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_n	0	0	1	$\cdots$	1

Finally, the data owner constructs the index according to $F(w_i)$. We use $v(w_i)$ to represent the index vector of keyword w_i and set $addr_i = w_i$. If the file $F_j(1 \le j \le N)$ is in $F(w_i)$, we set $v(w_i)[j] = 1$; otherwise, set $v(w_i)[j] = 0$.

4) SECURE INDEX

The data owner encrypts the plain index into the secure index, as shown in Table.3. He employs PRP π to confuse the real location of keywords, and employs PRF f to blind the vector $v(w_i)$, that is, $\pi_{k_1}(w_i)$ and $Ev(w_i) \leftarrow f_{k_2}(\pi_{k_1}(w_i)) \oplus v(w_i)$. The data owner invokes algorithm SKE.*Enc* to encrypt the document F into the cipher-text C. Use $ID(C_j)$ to denote the identifier of encrypted file C_j.

TABLE 3. Secure index.

W \ C	$ID(C_1)$	$ID(C_2)$	$ID(C_3)$	$\cdots$	$ID(C_N)$
$\pi_{k_1}(w_n)$	1	0	0	$\cdots$	0
$\pi_{k_1}(w_3)$	1	1	0	$\cdots$	1
$\pi_{k_1}(w_1)$	1	1	1	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\pi_{k_1}(w_2)$	1	0	1	$\cdots$	0

5) EDIT DISTANCE

Edit distance(Levenshtein) is used to evaluate the similarity of two strings in fuzzy keyword search scheme. The edit distance $ed(w_1, w_2)$ between two keywords 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: Transfer a character into another in a word.
2) Deletion: Delete a letter from a word.
3) Insertion: Insert a letter into a word .

For example, the operations required to transform the keyword *kitten* to the keyword *sitting* are as follows:

1) *kitten* $\rightarrow$ *sitten* ($k \rightarrow s$)
2) *sitten* $\rightarrow$ *sittin* ($e \rightarrow i$)
3) *sittin* $\rightarrow$ *sitting* (insert g)

Generally, the smaller the editing distance is, the greater the similarity of two keywords is.

E. DEFINITIONS

1) SCHEME DEFINITION

Definition 1 (Verifiable SSE Scheme): A verifiable SSE scheme includes six polynomial-time algorithms (KeyGen, BuildIndex, Trapdoor, Search, Verify and Dec).

- $K \leftarrow$ **KeyGen**(1^k) is the probabilistic key generation algorithm run by the data owner. It takes a random secure parameter k as input, and outputs a secret key set K.
- $(\mathcal{I}, \mathcal{C}) \leftarrow$ **BuildIndex**$(K, \mathcal{F})$ is the probabilistic index building algorithm run by the data owner. It takes the secret key set K and the file set $\mathcal{F}$ as input, and outputs a secure index $\mathcal{I}$ and a set of ciphertexts $\mathcal{C}$.
- $T_w \leftarrow$ **Trapdoor**(K, w) is the deterministic trapdoor generation algorithm run by the data user. It takes the secret key set K and the queried keyword w as input, and outputs the trapdoor T_w.
- $(tag_w, C(w)) \leftarrow$ **Search**$(T_w, \mathcal{I}, \mathcal{C})$ is the deterministic search algorithm run by the cloud server. It takes the trapdoor T_w, the secure index $\mathcal{I}$ and the ciphertexts set $\mathcal{C}$ as input, and outputs a set $C(w)$ and an authentication label tag_w.
- $(accept, reject) \leftarrow$ **Verify**$(K, T_w, C(w), tag_w)$ is the deterministic verification algorithm run by the data user. It takes the secret key set K, the trapdoor T_w, the set $C(w)$ and the authentication label tag_w as input, and outputs *accept* or *reject*.
- $F(w) \leftarrow$ **Dec**$(K, C(w))$ is the deterministic decryption algorithm run by the data user. It takes the secret key set K and the set $C(w)$ as input, and outputs a set of plaintext documents $F(w)$.

2) SECURITY DEFINITIONS

Definition 2 (Privacy): A Verifiable SSE scheme satisfies privacy if there exists a probabilistic polynomial-time *PPT* simulator **Sim** such that

$$|Pr(\mathcal{A} \text{ outputs } b = 1 \text{ in } Game_{real}) - Pr(\mathcal{A} \text{ outputs } b = 1 \text{ in } Game_{sim})|$$

is negligible for any *PPT* adversary $\mathcal{A}$.

We regard a real game Game_{real} and a simulation game Game_{sim}, in which Game_{real} is performed by a challenger and an adversary $\mathcal{A}$, and Game_{sim} is performed by a simulator **Sim**.

Real Game(Game_{real})

- $\mathcal{A}$ selects a pair of document set and keyword set $(\mathcal{F}, \mathcal{W})$, and sends them to the challenger.
- The challenger computes the secret key set K and $(\mathcal{I}, \mathcal{C})$ by calling algorithm **KeyGen** and algorithm **BuildIndex**, and sends $(\mathcal{I}, \mathcal{C})$ to $\mathcal{A}$.
- $\mathcal{A}$ chooses a keyword w and sends it to the challenger. The challenger sends the trapdoor T_w to $\mathcal{A}$.
- $\mathcal{A}$ outputs a bit b.

Simulation Game(Game_{sim})

- $\mathcal{A}$ chooses $(\mathcal{F}, \mathcal{W})$ and sends them to the challenger.
- The challenger sends $|F_1|, \ldots, |F_N|$ and l to **Sim**.
- **Sim** computes $(\mathcal{I}', \mathcal{C}')$ and sends them to the challenger.
- The challenger sends $(\mathcal{I}', \mathcal{C}')$ to $\mathcal{A}$.
  - $\mathcal{A}$ chooses w and sends it to the challenger.
  - **Sim** computes T'_w using PRP function π and PRF function f, and sends it to the challenger.

– The challenger sends T'_w to $\mathcal{A}$.

- $\mathcal{A}$ outputs a bit b.

Definition 3 (Correctness): A verifiable SSE scheme is correct if it satisfies the following requirements: $\forall k \in N, K \leftarrow$**KenGen**$(1^k), \forall F_i \in \mathcal{F}(1 \leq i \leq N), \forall w \in \mathcal{W}$, the following statement is true:

$$Search(\mathcal{I}, \mathcal{C}, T_w) = (C(w), tag_w) \wedge$$
$$Verify(K, (C(w), T_w, tag_w) = accept \wedge$$
$$Dec(K, C(w)) = F(w).$$

Definition 4 (Reliability): A verifiable SSE scheme satisfies reliability if for any PPT adversary $\mathcal{A}$, the probability of successfully forging the search result is negligible for any $(\mathcal{F}, \mathcal{W}, \mathcal{I})$ and trapdoor T_w.

Specifically, because the cloud server is untrusted, it may return incorrect search result to the data user. The data user should be able to detect such misbehaviour to guarantee the validity of search result. Given a valid $C(w)$ and an authentication label tag_w for a trapdoor, the adversary wins if he can give a forgery $(C'(w), tag'_w)$ that can pass the **Verify** algorithm.

IV. THE PROPOSED VERIFIABLE SEARCHABLE SYMMETRIC ENCRYPTION SCHEMES

A. THE PROPOSED VERIFIABLE EXACT KEYWORD SEARCH (VEKS) SCHEME

The algorithms of this scheme are described as follows.

1) **KeyGen**(1^k)
The data owner calls algorithm SKE.*Gen* to generate sk, that is, $sk \leftarrow$ SKE.*Gen*(1^k). Then he samples $k_0, k_1, k_2 \leftarrow \{0,1\}^k$. sk is the key for encrypting and decrypting documents, k_0 is the key for MAC, k_1 is the key for PRP π, and k_2 is the key for PRF f. The algorithm outputs the secret key set $K = \{sk, k_0, k_1, k_2\}$.

2) **BuildIndex**$(K, \mathcal{F})$
The process of building the secure index is shown in Fig.2.
   a) The data owner scans the whole documents to extract all of the distinct keywords and constructs the keyword set $\mathcal{W}$. For each keyword $w_i \in \mathcal{W}(1 \leq i \leq n)$, the data owner builds $F(w_i)$, which is a set of files containing w_i.
   b) According to $F(w_i)$, the data owner builds the index shown in Table.2. Let $v(w_i)(1 \leq i \leq n)$ represent the index vector of keyword w_i. If F_j $(1 \leq j \leq N)$ is in $F(w_i)$, the data owner sets $v(w_i)[j] = 1$; otherwise, sets $v(w_i)[j] = 0$.
   c) The data owner encrypts the documents by computing $C(w_i) \leftarrow$ SKE.$Enc_{sk}(F(w_i))$. Then he generates the keyword trapdoor $\pi_{k_1}(w_i)$ and blinds the vector $v(w_i)$ by computing $Ev(w_i) \leftarrow f_{k_2}(\pi_{k_1}(w_i)) \oplus v(w_i)$.
   d) The data owner computes $tag_{w_i} \leftarrow \text{MAC}(\pi_{k_1}(w_i), Ev(w_i), C(w_i))$ for each w_i, and stores all tag_{w_i} into the *Tag* table.

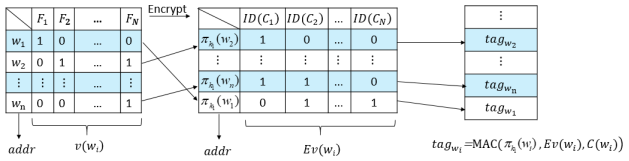


FIGURE 2. The process of building the secure index and the *Tag* table.

   e) Finally, the data owner uploads the secure index $\mathcal{I}$, the ciphertexts $\mathcal{C}$ and the *Tag* table to the cloud server.

3) **Trapdoor**(K, w')
When the data user wants to search some documents containing the keyword w', he computes the trapdoor $T_{w'} = (\pi_{k_1}(w'), f_{k_2}(\pi_{k_1}(w')))$, and sends it to the cloud server.

4) **Search**$(T_{w'}, \mathcal{C}, \mathcal{I})$
The cloud server finds the related encrypted vector $Ev(w')$ in the secure index $\mathcal{I}$ according to $\pi_{k_1}(w')$. Then it decrypts $Ev(w')$ by computing $v(w') \leftarrow f_{k_2}(\pi_{k_1}(w')) \oplus Ev(w')$. If $v(w')[j] = 1 (1 \leq j \leq N)$, the cloud server adds C_j to $C(w')$; otherwise, not. Meanwhile, the cloud server extracts $tag_{w'}$ on the $\pi_{k_1}(w')$-th position in the *Tag* table. Finally, the cloud server returns $C(w')$ and $tag_{w'}$ to the data user.

5) **Verify**$(K, tag_{w'}, C(w'), T_{w'})$
The data user parses out $v(w')$ from $C(w')$. If C_j is in $C(w')$, then the j-th bit in $v(w')$ is 1; otherwise, is 0. The data user computes $Ev(w') \leftarrow f_{k_2}(\pi_{k_1}(w')) \oplus v(w')$. Then he checks whether $\text{MAC}(\pi_{k_1}(w'), Ev(w'), C(w'))$ $= tag_{w'}$ or not. If it does, the data user accepts the result; otherwise, rejects the result.

6) **Dec**$(K, C(w'))$
The data user decrypts $C(w')$ by computing $F(w') \leftarrow$ SKE.$Dec_{sk}(C(w'))$ $(sk \in K)$.

B. THE PROPOSED VERIFIABLE FUZZY KEYWORD SEARCH (VFKS) SCHEME

When the data user makes spelling errors, the above VEKS scheme cannot return the files of interest. To address this problem, we extend the above VEKS scheme to a VFKS scheme. The detailed scheme is described as follows.

1) **KeyGen**(1^k)
It is the same as the key generation algorithm **KeyGen**(1^k) in section IV.A. The secret key set is $K = \{sk, k_0, k_1, k_2\}$.

2) **BuildIndex**$(K, \mathcal{F})$
In order to conveniently and efficiently build the index, we construct the linked lists containing three nodes as the index by extending the inverted index. The structure of this index is shown in Fig.3.
   a) The data owner scans the whole documents to extract all of the distinct keywords and constructs the keyword set $\mathcal{W}$. For each keyword $w_i \in \mathcal{W}$, the data owner builds $F(w_i)$, which is a set of files containing w_i.

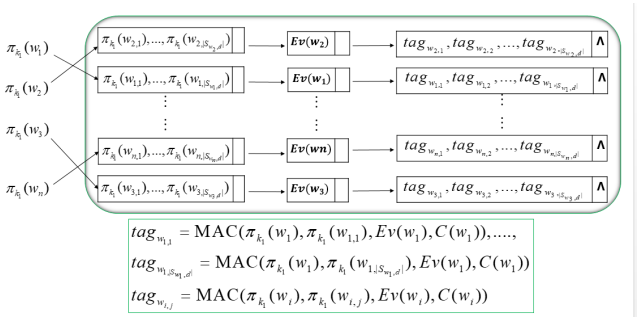


FIGURE 3. The structure of the secure index.

b) According to $F(w_i)$, the data owner builds the index shown in Table.2. Let $v(w_i)(1 \leq i \leq n)$ represent the index vector of keyword w_i. If F_j $(1 \leq j \leq N)$ is in $F(w_i)$, the data owner sets $v(w_i)[j] = 1$; otherwise, sets $v(w_i)[j] = 0$.

c) The data owner constructs the fuzzy keyword set for each keyword w_i. Let $S_{w_i,d}$ denote the fuzzy keyword set of w_i with edit distance d, and $w_{i,t}(1 \leq i \leq n, 1 \leq t \leq |S_{w_i,d}|)$ denote the keywords in $S_{w_i,d}$.

d) The data owner encrypts all of the files by computing $C(w_i) \leftarrow \text{SKE}.Enc_{sk}(F(w_i))$. He calculates $\pi_{k_1}(w_{i,t})(1 \leq i \leq n, 1 \leq t \leq |S_{w_i,d}|)$ for each fuzzy keyword in $S_{w_i,d}$, and stores them into the first node. The data owner computes $Ev(w_i) \leftarrow f_{k_2}(\pi_{k_1}(w_i)) \oplus v(w_i)$, and stores $Ev(w_i)$ into the second node. He computes an authentication label $tag_{w_{i,t}} = \text{MAC}(\pi_{k_1}(w_i), \pi_{k_1}(w_{i,t}), Ev(w_i), C(w_i))$ for all fuzzy keywords in $S_{w_i,d}$, and stores them into the third node.

e) Finally, the data owner stores these linked lists into a big array, which is regarded as the secure index $\mathcal{I}$.

3) **Trapdoor**(K, w')
When the data user wants to search some files containing the keyword w', he computes the trapdoor $T_{w'} = (\pi_{k_1}(w'), f_{k_2}(\pi_{k_1}(w')))$, and sends it to the cloud server.

4) **Search**$(T_{w'}, \mathcal{I}, \mathcal{C})$
When the cloud server receives the trapdoor, it compares $\pi_{k_1}(w')$ with the elements in the first node of each list. Here, we use encrypted exact keyword $\pi_{k_1}(w_i)$ to denote the first element in the first node $\pi_{k_1}(w_{i,1})$. That is, $w_{i,1}$ denotes the exact keyword w_i. As shown in Algorithm 1, we describe this process in two cases.
Case 1: $\pi_{k_1}(w')$ does not equal the first element $\pi_{k_1}(w_{i,1})$, but equals a remaining element $\pi_{k_1}(w_{i,t})(1 < t \leq |S_{w_i,d}|)$ in the first node. In this case, the cloud server returns $\pi_{k_1}(w_{i,1})$ to the data user. When the data user receives $\pi_{k_1}(w_{i,1})$, he calculates $f_{k_2}(\pi_{k_1}(w_{i,1}))$, and sends it to the cloud server. The cloud server decrypts $Ev(w_i)$ in the second node by computing $v(w_i) \leftarrow f_{k_2}(\pi_{k_1}(w_{i,1})) \oplus Ev(w_i)$. If $v(w_i)[j] = 1$, the cloud server adds C_j to $C(w')$; otherwise, not. Then the cloud server extracts $tag_{w'}$ from the $\pi_{k_1}(w')$-th position in the third node. Finally, the cloud server returns $C(w')$ and $tag_{w'}$ to the data user.
Case 2: $\pi_{k_1}(w')$ equals the first element $\pi_{k_1}(w_{i,1})$. In this case, the cloud server directly decrypts $Ev(w_i)$ in the second node by computing $v(w_i) \leftarrow f_{k_2}(\pi_{k_1}(w')) \oplus Ev(w_i)$. If $v(w_i)[j] = 1$, the cloud server adds C_j to $C(w')$; otherwise, not. Then the cloud server extracts $tag_{w'}$ from the $\pi_{k_1}(w')$-th position in the third node. Finally, the cloud server returns $C(w')$, $tag_{w'}$ and $\pi_{k_1}(w_{i,1})$ to the data user.

5) **Verify**$(K, tag_{w'}, C(w'), \pi_{k_1}(w_{i,1}), T_{w'})$
The data user extracts $v(w')$ from $C(w')$. If C_j is in $C(w')$, then the j-th bit in $v(w)$ is 1; otherwise, is 0. The data user computesi $Ev(w') \leftarrow f_{k_2}(\pi_{k_1}(w')) \oplus v(w')$. Then he checks whether

Algorithm 1 Searching Operation

Input:
The trapdoor $T_{w'}$, the secure index $\mathcal{I}$, and the encrypted document collection $\mathcal{C}$;
Output:
The authentication label $tag_{w'}$ and the document collection $C(w')$;

```
Parse T_{w'} as (η, θ);
for i ← 1 to n do
  for t ← 1 to |S_{w_i,d}| do
    if π_{k_1}(w_{i,t}) equals η and t ≠ 1 then
      Give π_{k_1}(w_{i,1}) to the data user, and the data user
      computes f_{k_2}(π_{k_1}(w_{i,1}));
      Sends f_{k_2}(π_{k_1}(w_{i,1})) to the cloud server;
      Extract Ev(w_i) from the second node and decrypt
      it to v(w_i) with f_{k_2}(π_{k_1}(w_{i,1}));
      for j ← 1 to N do
        if v(w_j)[j] = 1 then
          Add C_j to C(w');
        end if
      end for
      Extract tag_{w'} according to η from the third node;
      return tag_{w'} and C(w');
    end if
    if π_{k_1}(w_{i,t}) equals η and t = 1 then
      Extract Ev(w_i) and decrypt it to v(w_i) with θ;
      for j ← 1 to N do
        if v(w_i)[j] = 1 then
          Add C_j to C(w');
        end if
      end for
      Extract tag_{w'} according to η from the third node;
      return tag_{w'}, C(w') and π_{k_1}(w_{i,1});
    end if
  end for
end for
```

$\text{MAC}(\pi_{k_1}(w_{i,1}), \pi_{k_1}(w'), Ev(w'), C(w')) = tag_{w'}$ or not. If it does, the data user accepts the result; otherwise, rejects the result.

6) **Dec**$(sk, C(w'))$
The data user decrypts $C(w')$ by computing $F(w') \leftarrow \text{SKE}.Dec_{sk}(C(w'))$ $(sk \in K)$.

V. SECURITY ANALYSIS

In this section, we analyze the security of our proposed schemes in terms of the privacy, the correctness and the reliability.

A. SECURITY ANALYSIS OF THE VERIFIABLE EXACT KEYWORD SEARCH SCHEME

Theorem 1: The proposed VEKS scheme satisfies privacy.

Proof: Assume there exists a probabilistic polynomial time(PPT) simulator **Sim** and an adversary $\mathcal{A}$. In the phases of **KeyGen** and **BuildIndex**, we give the keyword number of each maximum document and the max length of keywords l to **Sim**.

1) **Sim** invokes algorithm **KeyGen** to generate the key set $K = \{sk, k_0, k_1, k_2\}$.
2) **Sim** computes $C'_i = Enc(sk, F_i)$ for $1 \le i \le N$ and $\mathcal{C}' = \{C'_1, .., C'_N\}$.
3) **Sim** constructs the secure index $\mathcal{I}'$ and generates the Tag' table for each keyword $w \in \mathcal{W}$:

$$\mathcal{I}'(\pi_{k_1}(w)) = Ev(w),$$

$$Tag'(\pi_{k_1}(w)) = tag_w,$$

where $tag_w = \text{MAC}(\pi_{k_1}(w), Ev(w), C(w))$ and $Ev(w) = v(w) \oplus f_{k_2}(\pi_{k_1}(w))$.

4) **Sim** returns $(\mathcal{I}', \mathcal{C}')$ to $\mathcal{A}$.
In the search phase, **Sim** computes $T'_{w'} = (\pi_{k_1}(w'), f_{k_2}(\pi_{k_1}(w')))$ and extracts the documents set $C(w') = \{C_j\}$ that contains the queried keyword w'.
We will prove that any adversary $\mathcal{A}$ cannot distinguish Game_{sim} and Game_{real} by using a series of games $\text{Game}_0, \text{Game}_1, \text{Game}_2$, where $\text{Game}_0 = \text{Game}_{real}$. Let

$$p_i = Pr(\mathcal{A} \text{ outputs } b = 1 \text{ in } \text{Game}_i).$$

- Game_1 is the same as Game_0 except that C_j is replaced by C'_j. Then $|p_0 - p_1|$ is negligible.
- Game_2 is the same as Game_1 except that $\mathcal{I}$ is replaced by $\mathcal{I}'$, Tag is replaced by Tag', and $T_{w'}$ is replaced by $T'_{w'}$. Let $addr_i = \pi_{k_1}(w')$. From above analysis, we can see that

$$\mathcal{I}'(addr_i) = \mathcal{I}'(\pi_{k_1}(w')) = Ev(w'),$$
$$Tag'(addr_i) = Tag'(\pi_{k_1}(w')) = tag_{w'}.$$

The values of $\mathcal{I}'(addr_i)$ and $Tag'(addr_i)$ are the same as the real values. Hence $|p_1 - p_2|$ is negligible.

Therefore, $|p_0 - p_2|$ is negligible. It is clear that $\text{Game}_2 = \text{Game}_{sim}$. It means that any adversary $\mathcal{A}$ cannot distinguish Game_{real} and Game_{sim}. Consequently, the proposed VEKS scheme satisfies privacy.

Theorem 2: The proposed VEKS scheme is correct.

Proof: Given the trapdoor $T_{w'} = (\pi_{k_1}(w'), f_{k_2}(\pi_{k_1}(w')))$ of the queried keyword w', the cloud server can locate the $\pi_{k_1}(w')$-th entry of the secure index and decrypt the vector by computing $v(w') \leftarrow f_{k_2}(\pi_{k_1}(w')) \oplus Ev(w')$. Then, the cloud server finds the related cipher documents according to the vector. If the bit of the vector is 1, it adds the relevant cipher document to the set $C(w')$. The cloud server extracts $tag_{w'}$, and sends $C(w')$ and $tag_{w'}$ to the data user. After receiving the search result from the cloud server, the data user extracts $v(w')$ and computes $Ev(w') \leftarrow f_{k_2}(\pi_{k_1}(w')) \oplus v(w')$. Finally, the data user checks whether $\text{MAC}(\pi_{k_1}(w'), Ev(w'), C(w')) = tag_{w'}$ or not. If the cloud server is honest and reliable, the search result will pass the verification. The data user can decrypt the documents if the result is right. Therefore, the proposed VEKS scheme is correct.

Theorem 3: The proposed VEKS scheme satisfies reliability.

Proof: Assume that $\mathcal{A}$ is a PPT adversary who can give a forgery $(C'(w'), tag'_{w'})$ such that the verification algorithm $Verify(K, tag'_{w'}, C'(w'), T_{w'})$ outputs $accept$. If such an adversary $\mathcal{A}$ exists, it will break the reliability of VEKS scheme. Suppose $(C(w'), tag_{w'})$ is the correct search result. We need to prove that the adversary $\mathcal{A}$ cannot forge a valid $(C'(w'), tag'_{w'})$ such that $(C'(w'), tag'_{w'}) = (C(w'), tag_{w'})$.

We need to consider two cases:

Case 1: $C'(w') \neq C(w')$. The adversary $\mathcal{A}$ computes $tag'_{w'} \leftarrow \text{MAC}(\pi_{k_1}(w'), Ev(w'), C'(w'))$ for the trapdoor $T_{w'}$ by replacing $C(w')$ with $C'(w')$, where $Ev(w')$ is the index vector.

Case 2: $C'(w') = C(w')$. As same as Case 1, the adversary $\mathcal{A}$ computes $tag'_{w'}$ for the trapdoor $T_{w'}$ by invoking $\text{MAC}(\pi_{k_1}(w'), Ev(w'), C'(w'))$.

In above cases, if the adversary $\mathcal{A}$ forges the authentication label $tag'_{w'}$ according to the method that we describe, the verification algorithm will output *"accept"*. $\mathcal{A}$ will trick the data user successfully and make him believe $C'(w')$ is valid. However, MAC provides irreversibility and message unforgeability against chosen message attacks, and $\mathcal{A}$ does not know the keys of MAC. The probability of forging a MAC output successfully for $\mathcal{A}$ is negligible. Thus, whether $C'(w') = C(w')$, $tag'_{w'} \neq tag_{w'}$, that is, $Verify(K, tag'_{w'}, C'(w'), T_{w'}) \neq accept$. In other words, $\mathcal{A}$ cannot break the reliability of VEKS scheme. Therefore, our proposed VEKS scheme satisfies the reliability.

B. SECURITY ANALYSIS OF THE VERIFIABLE FUZZY KEYWORD SEARCH SCHEME

Theorem 4: The proposed VFKS scheme satisfies privacy.

Proof: Assume there exists a probabilistic polynomial time(PPT) simulator **Sim** and an adversary $\mathcal{A}$. In the phases

of **KeyGen** and **BuildIndex**, we give the keyword number of each document and the max length of keywords l to **Sim**.

1) **Sim** invokes algorithm **KeyGen** to generate the key set $K = \{sk, k_0, k_1, k_2\}$.
2) **Sim** computes $C'_i = Enc(sk, F_i)$ for $1 \le i \le N$ and $\mathcal{C}' = \{C'_1, .., C'_N\}$.
3) **Sim** constructs the secure index $\mathcal{I}'$ containing three nodes. The first node stores the fuzzy keywords $\{\pi_{k_1}(w_{i,1}), \ldots, \pi_{k_1}(w_{i,q})\}$ of w_i, where q equals $|S_{w_i,d}|$, the second node stores the index vector $Ev(w_i)$, and the third node stores the authentication labels $\{tag_{w_{i,1}}, \ldots, tag_{w_{i,q}}\}$, where $tag_{w_{i,j}}$ =MAC$(\pi_{k_1}(w_{i,j}), \pi_{k_1}(w_i), Ev(w_i), C'(w_i)$, and $Ev(w_i) = v(w_i) \oplus f_{k_2}(\pi_{k_1}(w_i))$.
4) **Sim** returns $(\mathcal{I}', \mathcal{C}')$ to $\mathcal{A}$.

In the search phase, **Sim** computes $T'_{w'} = (\pi_{k_1}(w'), f_{k_2}(\pi_{k_1}(w')))$ and extracts the documents set $C(w') = \{C_j\}$ that contains the queried keyword w'.

We will prove that any adversary $\mathcal{A}$ cannot distinguish Game$_{sim}$ and Game$_{real}$ by using a series of games Game$_0$, Game$_1$, Game$_2$, where Game$_0$ = Game$_{real}$. Let

$$p_i = Pr(\mathcal{A} outputs\ b = 1\ in\ \text{Game}_i).$$

- Game$_1$ is the same as Game$_0$ except that C_j is replaced by C'_j. Then $|p_0 - p_1|$ is negligible.
- Game$_2$ is the same as Game$_1$ except that $\mathcal{I}$ is replaced by $\mathcal{I}'$, and $T_{w'}$ is replaced by $T'_{w'}$. Nextly, let $addr = \pi_{k_1}(w')$. From above analysis, we can see that

$$\mathcal{I}'_1(addr) = \pi_{k_1}(w_{i,j}),$$
$$\mathcal{I}'_2 = Ev(w_i),$$
$$\mathcal{I}'_3(addr) = tag_{w_{i,j}},$$

where $\mathcal{I}'_j$ denotes the j-th node in one linked list. The values of $\mathcal{I}'_2$ and $\mathcal{I}'_3(addr)$ are the same as the real values. Hence $|p_1 - p_2|$ is negligible.

Therefore, $|p_0 - p_2|$ is negligible. It is clear that Game$_2$ = Game$_{sim}$. It means that any adversary $\mathcal{A}$ cannot distinguish Game$_{real}$ and Game$_{sim}$. Consequently, the proposed VFKS scheme satisfies privacy.

Theorem 5: The proposed VFKS scheme is correct.

Proof: The proposed VFKS scheme can return the documents of interest when the data user makes spelling errors. When the cloud server receives the trapdoor, it compares $\pi_{k_1}(w')$ with the elements in the first node of each list. We use encrypted exact keyword $\pi_{k_1}(w_i)$ to denote the first element in the first node $\pi_{k_1}(w_{i,1})$. That is, $w_{i,1}$ denotes the exact keyword w_i.

Case 1: $\pi_{k_1}(w')$ does not equal the first element $\pi_{k_1}(w_{i,1})$, but equals a remaining element $\pi_{k_1}(w_{i,t})(1 < t \le |S_{w_i,d}|)$ in the first node. In this case, the cloud server returns $\pi_{k_1}(w_{i,1})$ to the data user. When the data user receives $\pi_{k_1}(w_{i,1})$, he computes $f_{k_2}(\pi_{k_1}(w_{i,1}))$, and sends it to the cloud server. The cloud server decrypts $Ev(w_i)$ in the second node by computing $v(w_i) \leftarrow f_{k_2}(\pi_{k_1}(w_{i,1})) \oplus Ev(w_i)$. If $v(w_i)[j] = 1$, the cloud server adds C_j to $C(w')$; otherwise, not. Then the cloud server extracts $tag_{w'}$ from the $\pi_{k_1}(w')$-th position in the third node. Finally the cloud server returns $C(w')$ and $tag_{w'}$ to the data user.

Case 2: $\pi_{k_1}(w')$ equals the first element $\pi_{k_1}(w_{i,1})$. In this case, the cloud server directly decrypts $Ev(w_i)$ in the second node by computing $v(w_i) \leftarrow f_{k_2}(\pi_{k_1}(w')) \oplus Ev(w_i)$. If $v(w_i)[j] = 1$, the cloud server adds C_j to $C(w')$; otherwise, not. Then the cloud server extracts $tag_{w'}$ from the $\pi_{k_1}(w')$-th position in the third node. Finally, the cloud server returns $C(w')$, $tag_{w'}$ and $\pi_{k_1}(w_{i,1})$ to the data user.

If the search result passes the **Verify** algorithm, the data user decrypts $C(w')$ to obtain the interested documents. Therefore, the proposed VFKS scheme is correct.

Theorem 6: The proposed VFKS scheme satisfies reliability.

Proof: Assume that $\mathcal{A}$ is a *PPT* adversary who can give a forgery $(C'(w'), tag'_{w'}, \pi'_{k_1}(w_{i,1}))$ such that the verification algorithm $Verify(K, tag'_{w'}, C'(w'), \pi'_{k_1}(w_{i,1}), T_{w'})$ outputs *accept*. If such an adversary $\mathcal{A}$ exists, it will break the reliability of VFKS scheme. Suppose $(C(w'), tag_{w'}, \pi_{k_1}(w_{i,1}))$ is the correct search result. We need to prove that the adversary $\mathcal{A}$ cannot forge a valid $(C'(w'), tag'_{w'}, \pi'_{k_1}(w_{i,1}))$ such that $(C'(w'), tag'_{w'}, \pi'_{k_1}(w_{i,1})) = (C(w'), tag_{w'}, \pi_{k_1}(w_{i,1}))$.

We need to consider four cases:

Case 1: $C'(w') \ne C(w')$ and $\pi'_{k_1}(w_{i,1}) \ne \pi_{k_1}(w_{i,1})$. The adversary $\mathcal{A}$ computes $tag'_{w'} \leftarrow$ MAC$(\pi_{k_1}(w'), \pi'_{k_1}(w_{i,1}), Ev(w'), C'(w'))$ for the trapdoor $T_{w'}$ by replacing $C(w')$ with $C'(w')$ and $\pi_{k_1}(w_{i,1})$ with $\pi'_{k_1}(w_{i,1})$, where $Ev(w')$ is the index vector.

Case 2: $C'(w') \ne C(w')$ and $\pi'_{k_1}(w_{i,1}) = \pi_{k_1}(w_{i,1})$. As same as Case 1, the adversary $\mathcal{A}$ computes $tag'_{w'} \leftarrow$ MAC$(\pi_{k_1}(w'), \pi'_{k_1}(w_{i,1}), Ev(w'), C'(w'))$.

Case 3: $C'(w') = C(w')$ and $\pi'_{k_1}(w_{i,1}) \ne \pi_{k_1}(w_{i,1})$. As same as Case 1, the adversary $\mathcal{A}$ computes $tag'_{w'} \leftarrow$ MAC$(\pi_{k_1}(w'), \pi'_{k_1}(w_{i,1}), Ev(w'), C'(w'))$.

Case 4: $C'(w') = C(w')$ and $\pi'_{k_1}(w_{i,1}) = \pi_{k_1}(w_{i,1})$. As same as Case 1, the adversary $\mathcal{A}$ computes $tag'_{w'} \leftarrow$ MAC$(\pi_{k_1}(w'), \pi'_{k_1}(w_{i,1}), Ev(w'), C'(w'))$.

In above cases, if the adversary $\mathcal{A}$ forges the authentication label $tag'_{w'}$, the verification algorithm will output *"accept"*. $\mathcal{A}$ will trick the data user successfully and make him believe $C'(w')$ is valid. However, MAC provides irreversibility and message unforgeability against chosen message attacks, and $\mathcal{A}$ does not know the keys of MAC. The probability of forging a MAC output successfully for $\mathcal{A}$ is negligible. Thus, whether $C'(w') = C(w')$ or $\pi'_{k_1}(w_{i,1}) = \pi_{k_1}(w_{i,1})$, $tag'_{w'} \ne tag_{w'}$, that is, $Verify(K, tag'_{w'}, C'(w'), \pi'_{k_1}(w_{i,1}), T_{w'}) \ne accept$. In other words, $\mathcal{A}$ cannot break the reliability of VFKS scheme. Therefore, our proposed VFKS scheme satisfies the reliability.

VI. PERFORMANCE COMPARISON AND EXPERIMENT EVALUATION

A. PERFORMANCE COMPARISON

In Table.4, we compare the functionality of our schemes with that of Kaoru's scheme [15], Li's scheme [16], Wang's scheme [17] and Zhu's scheme [18]. Our proposed VEKS scheme and the scheme [15] both can realize the exact keyword search and the verification for search result, but cannot realize the fuzzy keyword search. The scheme [16] can realize the fuzzy keyword search but cannot support the verification for search result. Our proposed VFKS scheme, the scheme [17] and the scheme [18] all can realize the fuzzy keyword search and verification for search result.

TABLE 4. Functionality comparison.

Properties	scheme [15]	scheme [16]	scheme [17]	scheme [18]	VEKS	VFKS
Exact keyword	√				√	
Fuzzy keyword		√	√	√		√
Verification	√		√	√	√	√

In Table.5, we perform the efficiency comparison in the following aspects: index building cost, trapdoor generation cost, search cost, and verification cost. We denote N as the number of documents, n as the number of exact keywords, l as the maximum length of exact keywords, m as the total number of fuzzy keywords, M as the maximum number of fuzzy keywords in the fuzzy keyword set $S_{w_i,d}$, and M' as the number of fuzzy keywords for the queried keyword.

In the index building phase, the scheme [15] needs to compute $2^l N$ PRPs and $2^l N$ MACs. In comparison, our VEKS scheme needs to compute n PRPs, n PRFs and n MACs. Therefore, the index building time complexity is $O(N)$ in scheme [15] and is $O(n)$ in our VEKS scheme. In our experiment described in section VI.B, we extract at most 5000 keywords from 10000 documents. The number of documents N is bigger than the number of keywords n. In consequence, the index building efficiency in our VEKS scheme is superior to that in the scheme [15]. The schemes [16], [17] both adopt the symbol-tree to build secure index. The time complexity of index building is $O(nM)$. The scheme [18] needs to compute $2m$ PRFs, while our VFKS scheme needs to computes n PRFs and n PRPs. Therefore, the time complexity of index building is $O(m)$ in scheme [18] and is $O(n)$ in our VFKS scheme. Since an exact keyword can generate many fuzzy keywords, the number of fuzzy keywords m is much larger than the number of exact keywords n. Hence, $O(m)$ is much bigger than $O(n)$. Obviously, the efficiency of index building in our VFKS scheme is higher than that in schemes [16]–[18].

In the trapdoor generation phase, the scheme [15] needs to compute N PRPs. The time complexity of trapdoor generation in this scheme is $O(N)$. The schemes [16]–[18] all generate the fuzzy keyword set for the queried keyword and compute at most M PRFs and M PRPs. Therefore, the time complexity of trapdoor generation is respectively $O(M)$. In comparison, our VEKS scheme and VFKS scheme only need to compute one PRF and one PRP, and the time complexity of trapdoor generation is only $O(1)$. It is clear that our schemes are more efficient than schemes [16]–[18] in the trapdoor generation phase.

In the search phase, the scheme [15] needs to search N times, while our VEKS scheme only needs once search. Therefore, the search time complexity is $O(N)$ in [15], and is only $O(1)$ in our VEKS scheme. The schemes [16], [17] all need to traverse the index tree for each fuzzy keyword of the queried keyword. The search time complexity is $O(M'h)$ (where h is the height of the index tree). The scheme [18] makes $M'm$ times comparison at most, while our VFKS scheme only needs to make m times comparison at most. Therefore, the search time complexity is $O(M'm)$ in [18], and is $O(m)$ in our VFKS scheme. Accordingly, the search efficiency in our VEKS scheme is superior to that in the scheme [15], and the search efficiency in our VFKS scheme is higher than that in the scheme [18]. Furthermore, since we do not generate the fuzzy keyword set for the queried keyword, the VFKS scheme reduces a large amount of computation on the user side.

In the verification phase, the scheme [15] needs to compute N MACs. The verification time complexity in this scheme is $O(N)$. Our VEKS scheme and VFKS scheme both only need to compute one MAC. The verification time complexity is respectively $O(1)$ in our two schemes. Therefore, our VEKS scheme is more efficient than the scheme [15] in verification. In the scheme [17], the verification time complexity is $O(1)$. The scheme [18] needs to compute two accumulators in the verification phase, and the verification time complexity is $O(1)$. However, the verification in this scheme is based on the public key system, so verification is time-consuming. As a consequence, our VFKS scheme is more efficient than scheme [18] in verification.

From the above analysis, we can know that our proposed schemes are more efficient than existing schemes in the phases of index building, trapdoor generation, search and verification.

B. EXPERIMENT EVALUATION

In order to show the efficiency of our proposed schemes, we conduct experiments on a real-world dataset. We select 10000 real data files from the online database [38]. We implement our experiments using C++ language on a Linux OS equipped with 3.4GHz Inter(R) Core(TM) i7-6700 CPU and 16GB RAM.

1) INDEX CONSTRUCTION

In the VEKS scheme, the secure index building phase contains two major steps: encrypting the keywords(PRP) and blinding the index vectors(PRF). The time cost of encrypting the keywords is related to the number of keywords, and the time cost of blinding the index vector is related to the number of documents. Fig. 4 shows that the time cost of building the secure index is almost linear with the number of documents

TABLE 5. Efficiency comparison.

Properties	scheme [15]	VEKS	scheme [16]	scheme [17]	scheme [18]	VFKS
Index building cost	$O(N)$	$O(n)$	$O(nM)$	$O(nM)$	$O(m)$	$O(n)$
Trapdoor generation cost	$O(N)$	$O(1)$	$O(M)$	$O(M)$	$O(M)$	$O(1)$
Search cost	$O(N)$	$O(1)$	$O(M'h)$	$O(M'h)$	$O(M'm)$	$O(m)$
Verification cost	$O(N)$	$O(1)$	–	$O(1)$	$O(1)$	$O(1)$

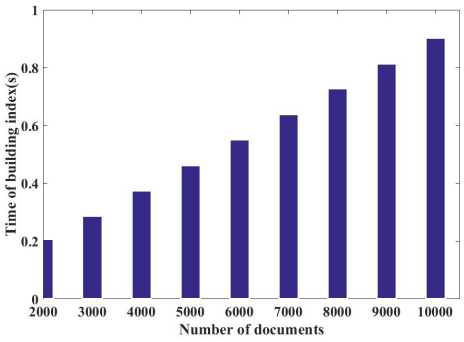


FIGURE 4. Index building time cost(VEKS)($n = 5000$).

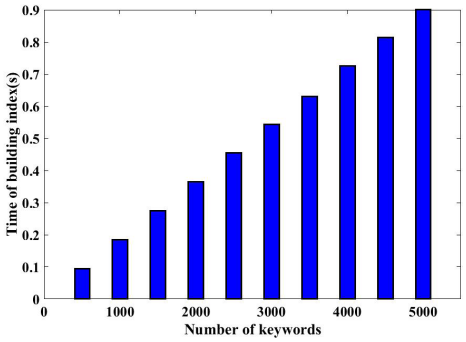


FIGURE 5. Index building time cost(VEKS)($N = 10000$).

in the experiment when the number of keywords n is fixed as 5000. Fig. 5 shows that the time cost of building the secure index is almost linear with the number of keywords in the experiment when the number of documents N is fixed as 10000.

In the VFKS scheme, we first need to construct the fuzzy keyword set for each keyword. Because the length of each keyword is different, the number of fuzzy keywords and the time of generating all fuzzy keywords are different. The longer the length of one keyword is, the higher the time cost of constructing the fuzzy keyword set is. Fig.6 shows the time cost of generating the fuzzy keyword set with edit distance $d = 1$. We can see that the time cost increases with the number of keywords. After all of the fuzzy keyword sets are constructed, we build the secure index for the VFKS scheme. The secure index building phase contains three major steps: encrypting the fuzzy keywords, blinding the index vectors, and generating the authentication labels for the fuzzy keywords. The time cost of building the secure index depends on the number of fuzzy keywords and the number of documents.

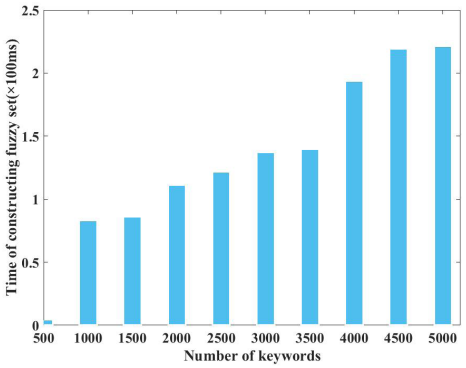


FIGURE 6. Fuzzy keyword set constructing time cost.

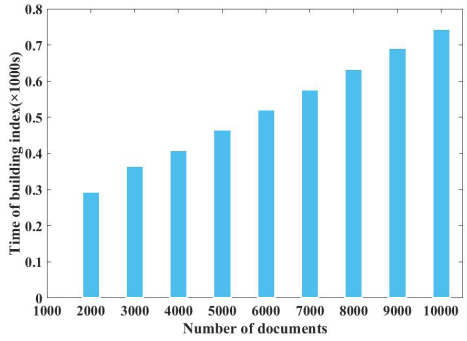


FIGURE 7. Index building time cost(VFKS)($n = 5000$).

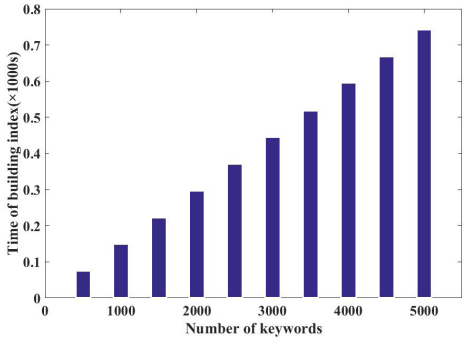


FIGURE 8. Index building time cost(VFKS)($N = 10000$).

Fig.7 shows that the time cost of building the secure index increases with the number of documents when the number of keywords n is fixed as 5000. Fig.8 shows that the time cost of building the secure index increases with the number of keywords when the number of documents N is fixed as 10000.

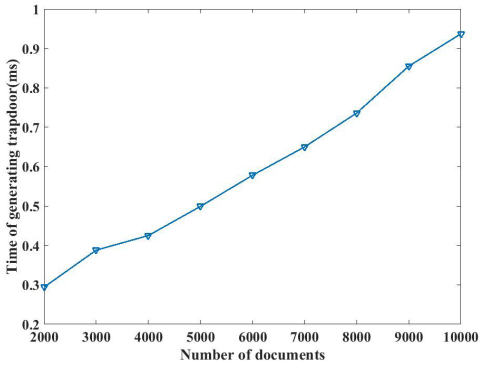


FIGURE 9. Trapdoor generation time cost.

2) TRAPDOOR GENERATION

In the VEKS scheme and the VFKS scheme, we adopt a two-tuple $T_w = (\pi_{k_1}(w), f_{k_2}(\pi_{k_1}(w)))$ as the trapdoor. The trapdoor only concludes two sequences of pseudo-random bits generated by PRP π and PRF f. The output length of π is a constant value, and the output length of f is related to the number of documents. As shown in Fig.9, the time cost of generating the trapdoor is linear with the number of documents.

3) SEARCH EFFICIENCY

In the VEKS scheme, the search phase contains two major steps: extracting the related index vector along with the authentication label and decrypting the vector to obtain the documents containing the queried keyword. The time of successfully matching the queried keyword with the keywords in the index is related to the number of keywords. As shown in Fig.10, the search time cost is almost linear with the number of keywords.

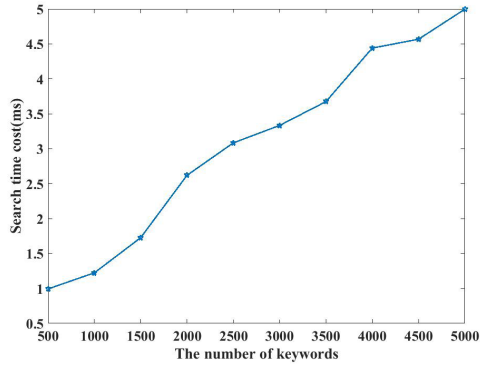


FIGURE 10. Search time cost(VEKS).

In the VFKS scheme, the search phase contains three major steps: comparing the trapdoor with the fuzzy keywords, extracting the related index vector along with the authentication label and decrypting the index vector to obtain the documents containing the queried keyword. The time cost of comparing the trapdoor with the fuzzy keywords is related to the number of fuzzy keywords. The time complexity of extracting the related index vector along with the authentication label is $O(1)$. The time cost of decrypting the index vector is related to the number of documents. Because the number of fuzzy keywords is much more than the number of documents, the number of fuzzy keywords has a greater impact on search time cost than the number of documents. Therefore, we only do the experiment to show the search time cost as the number of keywords grows. As shown in Fig.11, the search time cost is nearly linear with the number of keywords.

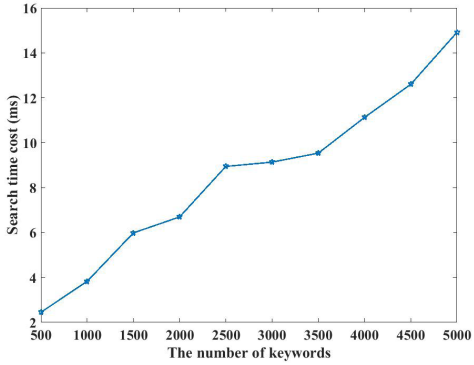


FIGURE 11. Search time cost(VFKS).

4) VERIFY EFFICIENCY

In the VEKS scheme, we need to verify whether $\text{MAC}(\pi_{k_1}(w), Ev(w), C(w))$ equals tag_w from the cloud server, where $C(w)$ is a set of documents containing the queried keyword, $Ev(w)$ is the encrypted index vector by computing $Ev(w) \leftarrow f_{k_2}(\pi_{k_1}(w)) \oplus v(w)$, and $v(w)$ is extracted from $C(w)$. The verification time complexity is $O(1)$. Fig.12 shows the verification time cost increases with the number of documents.

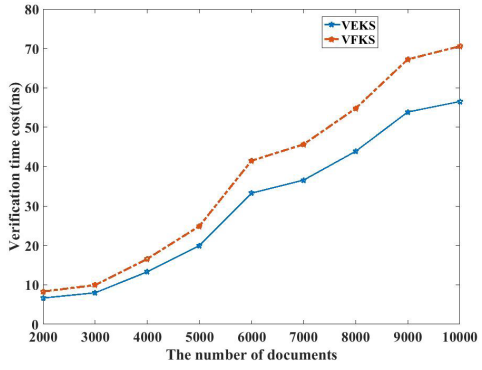


FIGURE 12. Verification time cost.

In the VFKS scheme, the verification process is the same as the VEKS scheme's. As shown in Fig.12, the verification time cost increases with the number of documents.

VII. DISCUSSION

In this section, we will discuss some problems and potential solutions. In the proposed VEKS and VFKS schemes, we do

not consider the problems of how to hide the number of keywords, how to hide the length of keywords, how to efficiently construct the fuzzy keyword set, and how to rank the returned documents. We will discuss the solutions to these problems as follows.

1) **How to hide the number of keywords.** To achieve the privacy of keywords, we can hide the number of keywords by padding the secure index in the VEKS scheme. Suppose each keyword is at most l bits and the number of keywords is at most 2^l, $n \leq 2^l$. The size of the secure index is n. We enlarge it to 2^l and initiate index by padding *dummy* and 0, as

$$addr = dummy$$
$$\mathcal{I}(dummy) = [\underbrace{0, \ldots, 0}_{N}]$$

Then we map the index vector $Ev(w)$ according to $\pi_{k_1}(w)$, as

$$addr = \pi_{k_1}(w)$$
$$\mathcal{I}(addr) = [\underbrace{1, 1, 0, \ldots 0, 1}_{N(Ev(w))}]$$

The remaining $2^l - n$ positions are still *dummy*. The vector 0 is encrypted by $f(dummy)$. The index with padding is shown in Fig.13. In this way, we can hide the real number of keywords from the malicious cloud server and the adversary.

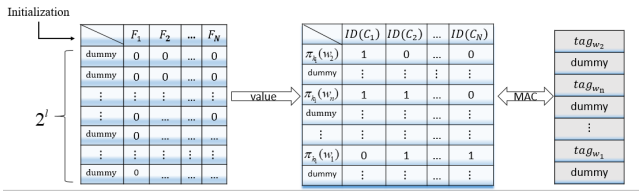


FIGURE 13. The index with padding.

2) **How to hide the length of keywords.** In the VFKS scheme, since the length of keywords is different, the number of fuzzy keywords in each fuzzy keyword set is different. The adversary and the malicious cloud server can learn the length of keyword according to the size of fuzzy keyword set. To address this problem, we can hide the size of each fuzzy keyword set by padding some random values. Firstly, we select the longest keyword and compute the number of its fuzzy keywords. Nextly, we pad other fuzzy keyword sets to the same size with the longest fuzzy keyword set. Finally, we pad the authentication label set at the same position with the fuzzy keyword set. In this way, we can hide the real length of each keyword from the malicious cloud server and the adversary.
3) **How to efficiently construct the fuzzy keyword set.** It needs large storage space to enumerate the fuzzy keywords, and results in low search efficiency. The number of keywords w' satisfying $ed(w, w') \leq 1$ is $(2l + 1) \times 26 + 1$. We can make use of the wildcard technology [16]–[18] to construct the fuzzy keyword set. A wildcard denotes all of the edit operations at the same position. For example, the fuzzy keyword set of the keyword *cat* with the edit distance $d = 1$ is $S_{cat,1} = \{cat, *cat, *at, c*at, c*t, ca*t, ca*, cat*\}$. In this case, the number of fuzzy keyword is 8, which is much smaller than the number of keywords 183 using enumeration method. For a keyword with the length of l bits, the number of fuzzy keyword is $2l + 2$, which is far less than $(2l + 1) \times 26 + 1$. This method helps us improve search efficiency and save the storage space. The detailed process of constructing the fuzzy keyword set is shown in Algorithm 2.

Algorithm 2 Fuzzy Keyword Set Construction

Input:
The keyword w_i, $1 \leq i \leq n$ and the edit distance d;
Output:
the fuzzy keyword set $S'_{w_i,d}$;

procedure FuzzySet(w_i, d)
if $d > 1$ **then**
  Call FuzzySet(w_i, $d - 1$);
end if
if $d = 0$ **then**
  Set $S'_{w_i,d} = \{w_i\}$;
else
  for $k \leftarrow 1$ to $|S'_{w_i,d-1}|$ **do**
    for $j \leftarrow 1$ to $2 * |S'_{w_i,d}[k]| + 1$ **do**
      if j is odd **then**
        Set fuzzykeyword as $S'_{w_i,d}[k]$;
        Insert $\star$ at position $\lfloor (j+1)/2 \rfloor$;
      else
        Set fuzzykeyword as $S'_{w_i,d}[k]$;
        Replace $\lfloor j/2 \rfloor - th$ character with $\star$;
      end if
      if fuzzykeyword is not in $S'_{w_i,d-1}$ **then**
        Set $S'_{w_i,d} = S'_{w_i,d-1} \cup \{\text{fuzzykeyword}\}$;
      end if
    end for
  end for
end if
return NONE;

4) **How to rank the documents.** The documents returned from the cloud server are not always what we are interested in. Because the relevance of the queried keyword with each document is different, it will cause unnecessary calculation cost if the cloud server returns all of the matched documents. In order to solve this problem, we can sort the documents by the relevance of the documents with the queried keyword. We use a widely-used statistic measurement similar to [29] to compute the relevance scores of matching documents to a search

trapdoor, named as TF×*IDF* rule. Here, TF denotes the number of a given keyword in a document, and *IDF* is obtained by dividing the cardinality of the document collection by the number of documents matching the queried keyword. The document is considered more relevant if it contains more matched query keywords. Simultaneously, we need to set a threshold T. The cloud server only returns the top-T documents to the data user. By utilizing this approach, the user can get the documents with higher interests, and meanwhile the system achieves better efficiency.

VIII. CONCLUSION

In this paper, we first propose a verifiable exact keyword search (VEKS) scheme over encrypted cloud data. And then construct a verifiable fuzzy keyword search (VFKS) scheme based on VEKS scheme. For enhancing the efficiency, we employ the linked lists as our secure index structure. Compared with the existing schemes, the proposed schemes achieve more efficient verification for search results. We give detailed security analysis of the proposed schemes. Comprehensive experiment evaluation indicates that they are very efficient.

REFERENCES

[1] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.

[2] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Trans. Depend. Sec. Comput.*, to be published, doi: 10.1109/TDSC.2018.2829880.

[3] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1362–1375, Jun. 2016.

[4] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Secur. Privacy Symp.*, May 2000, pp. 44–55.

[5] R. Curtmola, J. Gary, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.

[6] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Jan. 2016.

[7] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[8] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer-Verlag, 2013, pp. 353–373.

[9] J. Li, X. Chen, F. Xhafa, and L. Barolli, "Secure deduplication storage systems with keyword search," in *Proc. IEEE Int. Conf. Adv. Inf. Netw. Appl.*, May 2014, pp. 1532–1541.

[10] J. Li, X. Chen, F. Xhafa, and L. Barolli, "Secure deduplication storage systems supporting keyword search," *J. Comput. Syst. Sci.*, vol. 81, no. 8, pp. 1532–1541, 2015.

[11] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2010, pp. 253–262.

[12] W. Zhang, S. Xiao, Y. Lin, T. Zhou, and S. Zhou, "Secure ranked multi-keyword search for multiple data owners in cloud computing," in *Proc. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, Jun. 2014, pp. 276–286.

[13] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 829–837.

[14] J. Cui, H. Zhou, H. Zhong, and Y. Xu, "AKSER: Attribute-based keyword search with efficient revocation in cloud computing," *Inf. Sci.*, vol. 423, pp. 343–352, Jan. 2018.

[15] K. Kurosawa and Y. Ohtaki, "UC-secure searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2012, pp. 285–298.

[16] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. J. Lou, "Fuzzy keyword search over encrypted data in cloud computing," *Int. J. Eng. Res. Appl.*, vol. 4, no. 7, pp. 197–202, 2014.

[17] J. Wang *et al.*, "Efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *Comput. Sci. Inf. Syst.*, vol. 10, no. 2, pp. 667–684, 2013.

[18] X. Zhu, Q. Liu, and G. Wang, "A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2017, pp. 845–851.

[19] Z. Fu, X. Wu, Q. Wang, and K. Ren, "Enabling central keyword-based semantic extension search over encrypted outsourced data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 12, pp. 2986–2997, Dec. 2017.

[20] C. Liu, L. Zhu, and J. Chen, "Efficient searchable symmetric encryption for storing multiple source dynamic social data on cloud," *J. Netw. Comput. Appl.*, vol. 86, no. 3, pp. 3–14, 2017.

[21] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Proc. IEEE Int. Conf. Data Eng.*, Apr. 2012, pp. 1156–1167.

[22] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 2112–2120.

[23] J. Wang, X. Yu, and M. Zhao, "Privacy-preserving ranked multi-keyword fuzzy search on cloud encrypted data supporting range query," *Arabian J. Sci. Eng.*, vol. 40, no. 8, pp. 2375–2388, 2015.

[24] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Dec. 2016.

[25] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2012, pp. 917–922.

[26] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.

[27] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 2110–2118.

[28] J. Wang, X. Chen, J. Li, J. Zhao, and J. Shen, "Towards achieving flexible and verifiable search for outsourced database in cloud computing," *Future Gener. Comput. Syst.*, vol. 67, pp. 266–275, Feb. 2017.

[29] X. Jiang, J. Yu, J. Yan, and R. Hao, "Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data," *Inf. Sci.*, vols. 403–404, pp. 22–41, Sep. 2017.

[30] L. Chen and N. Zhang, "Efficient verifiable multi-user searchable symmetric encryption for encrypted data in the cloud," in *Proc. Int. Conf. Secur. Privacy New Comput. Environ.*, 2016, pp. 173–183.

[31] W. Zhang, Y. Lin, and Q. Gu, "Catch you if you misbehave: Ranked keyword search results verification in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 74–86, Jan./Mar. 2018.

[32] Y. Miao, J. Ma, X. Liu, J. Zhang, and Z. Liu, "VKSE-MO: Verifiable keyword search over encrypted data in multi-owner settings," *Sci. China Inf. Sci.*, vol. 60, no. 12, p. 122105, 2017.

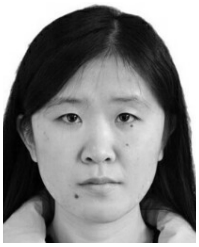
[33] Z. Fu, J. Shu, X. Sun, and N. Linge, "Smart cloud search services: Verifiable keyword-based semantic search over encrypted cloud data," *IEEE Trans. Consum. Electron.*, vol. 60, no. 4, pp. 762–770, Nov. 2014.

[34] J. Wang, X. Chen, and J. Li, "Verifiable search for dynamic outsourced database in cloud computing," in *Proc. Int. Conf. Broadband Wireless Comput.*, Nov. 2015, pp. 568–571.

[35] J. Yu, R. Hao, H. Xia, H. Zhang, X. Cheng, and F. Kong, "Intrusion-resilient identity-based signatures: Concrete scheme in the standard model and generic construction," *Inf. Sci.*, vols. 442–443, pp. 158–172, May 2018.

[36] Y. Xu, M. Wang, H. Zhang, J. Cui, L. Liu, and V. N. L. Franqueira, "Verifiable public key encryption scheme with equality test in 5G networks," *IEEE Access*, vol. 5, pp. 12702–12713, 2017.

[37] H. Krawczyk, "The order of encryption and authentication for protecting communications (or: How secure is SSL?)," in *Proc. Int. Conf. Adv. Cryptol.*, 2001, pp. 310–331.
[38] C. William. *Enron Email Dataset*. Accessed: Jun. 2008. [Online]. Available: https://www.cs.cmu.edu/~enron/



XINRUI GE received the B.E. degree in information security from Qingdao University in 2016. She is currently pursuing the master's degree in computer science and technology with Qingdao University. Her research interests include cloud computing security and searchable encryption.

JIA YU received the B.S. and M.S. degrees from the School of Computer Science and Technology, Shandong University, in 2000 and 2003, respectively, and the Ph.D. degree from the Institute of Network Security, Shandong University, in 2006. He was a Visiting Professor with the Department of Computer Science and Engineering, The State University of New York at Buffalo, Buffalo, NY, USA, from 2013 to 2014. He is currently a Professor with the College of Computer Science and Technology, Qingdao University. His research interests include cloud computing security, key evolving cryptography, digital signature, and network security.

CHENGYU HU received the Ph.D. degree from Shandong University in 2008. He is currently a Lecturer with the School of Software, Shandong University. His main research interests include cloud system security, public key cryptography, and leakage-resilient cryptography.

HANLIN ZHANG received the B.S. degree in software engineering from Qingdao University in 2010 and the M.S. degree in applied information technology and the Ph.D. degree in information technology from Towson University, Towson, MD, USA, in 2011 and 2016, respectively. He is currently with the College of Computer Science and Technology, Qingdao University, as an Assistant Professor. His research interests include information security, cloud security, mobile security, network security, cloud computing security, key evolving cryptography, digital signature, and network security.

RONG HAO received the master's degree from the Institute of Network Security, Shandong University. She is currently with the College of Computer Science and Technology, Qingdao University. Her research interest is information security.