IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 12, 2018, accepted April 11, 2018, date of publication April 20, 2018, date of current version May 9, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2828404

An Efficient Ranked Multi-Keyword Search for Multiple Data Owners Over Encrypted Cloud Data

TIANYUE PENG, (Student Member, IEEE), YAPING LIN, (Member, IEEE), XIN YAO, (Student Member, IEEE), AND WEI ZHANG

College of Computer Science and Electronic Engineering, Hunan University, Changsha 410006, China
Hunan Provincial Key Laboratory of Dependable Systems and Networks, Hunan University, Changsha 410012, China

Corresponding author: Yaping Lin (yplin@hnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Project 61472125 and in part by the Foundation of CERNET in China under Grant NGII20150407.

ABSTRACT With the development of cloud storage, more data owners are inclined to outsource their data to cloud services. For privacy concerns, sensitive data should be encrypted before outsourcing. There are various searchable encryption schemes to ensure data availability. However, the existing search schemes pay little attention to the efficiency of data users' queries, especially for the multi-owner scenario. In this paper, we proposed a tree-based ranked multi-keyword search scheme for multiple data owners. Specifically, by considering a large amount of data in the cloud, we utilize the TF $\times$ IDF model to develop a multi-keyword search and return the top-k ranked search results. To enable the cloud servers to perform a secure search without knowing any sensitive data (e.g., keywords and trapdoors), we construct a novel privacy-preserving search protocol based on the bilinear mapping. To achieve an efficient search, for each data owner, a tree-based index encrypted with an additive order and privacy-preserving function family is constructed. The cloud server can then merge these indexes effectively, using the depth-first search algorithm to find the corresponding files. Finally, the rigorous security analysis proves that our scheme is secure, and the performance analysis demonstrates its efficacy and efficiency.

INDEX TERMS Multi-keyword ranked search, multiple data owners, security, cloud storage.

I. INTRODUCTION

Cloud storage enables ubiquitous, scalable, and on-demand network access to a shared pool of digital data resources [1]. More enterprises and individuals tend to outsource their personal data to the cloud server, and utilize query services to easily access data anytime, anywhere and on any device. As one exemplary popular cloud storage services, Dropbox has 500 million users and 8 million business customers as of December 2017. The Cisco survey predicts that the global storage capacity would reach 1.1ZB, which is almost twice the space available in 2017. Besides, the "Cloud Storage Market by Solution (Primary Storage, Disaster Recovery & Backup Storage, Cloud Storage Gateway & Data Archiving), Service, Deployment Model (Public, Private & Hybrid), Organization Size, Vertical & Region - Global Forecast to 2021" reports that the cloud storage market is expected to grow from $23.76 billion in 2016 to $74.94 billion by 2021, and reach $97.41 billion by 2022.

Searchable symmetric encryption (SSE) [2]–[15] is often considered as a way to guarantee data privacy and data efficiency. However, various data owners encrypt their data with different keys leading to the following two drawbacks: (1) data users need to manage multiple keys for different data owners; (2) data users need to generate multiple trapdoors for data owners' data even for the same query condition. In this paper, we focus on *multiple data owners top-k query*, whereby the cloud server can merge multiple data indexes encrypted with different keys and efficiently support top-k query.

A. MOTIVATION

Data sharing is another crucial utility function, i.e., sharing data files with each other. In personal health record system, data user (e.g., a patient) should have the ability to access his/her top-k data files about a specific case from different data owners (e.g., health monitors, hospitals, doctors). Similarly, the employees in an enterprise should have the ability to search data files outsourced by other employees.

Recent work [16] proposed a privacy-preserving ranked multi-keyword search in a multi-user model (PRMSM),

which addresses the multi-keyword search problem in the multiple data owners model. However, PRMSM is inefficient and potentially expensive for frequent queries due to matching various ciphertexts from different data owners even for the same query.

B. CHALLENGE

In contrast to the single-user scenario, developing an efficient scheme for multiple data owners becomes a new challenge. To implement privacy preservation and efficient searches, we commonly build a tree-based index structure for each data owner's encrypted data. For a specific query condition, data users need to generate a trapdoor for each data owner, and the cloud should also search each index. This is obviously inefficient, due to the linear relationship of the number of trapdoors and data owners. A simple way to overcome this limitation is to let each data owner utilize the same key to encrypt their data files. Nevertheless, any one of the owners being compromised may lead to a system crash.

C. APPROACH

In this paper, we consider a multi-source cloud system, in which each data owner (viewed as a source) generates a tree-based index for his/her data files and encrypts these data with his/her corresponding key. To implement both privacy preservation and efficiency searches, we propose an efficient tree-based ranked multi-keyword search scheme (TBMSM). In this scheme, the cloud server is allowed to effectively merge multiple encrypted indexes, and securely perform the multi-keyword search without revealing the data owners' sensitive information, neither data files nor the queries. We construct a novel search protocol based on bilinear pairing, which enables different data owners to use different keys to encrypt their keywords and trapdoors. In order to rank the search results, we utilize the $TF \times IDF$ scheme to model relevance scores of data files and propose a ''Depth-First Search''(DFS) algorithm to obtain the ranked results. Finally, we confirm the security and efficiency of our scheme through comprehensive theoretical analysis and extensive experiments with a real dataset.

D. CONTRIBUTIONS

In summary, this paper makes the following contributions:

1) We construct a novel privacy-preserving search protocol, which allows the cloud server to perform an efficient secure multi-keyword ranked search without knowing data owners' sensitive information.
2) To achieve query efficiency, we introduce a consolidation strategy to implement multiple index trees. With this strategy, each data owner can encrypt their own tree-based index, and the cloud can be permitted to effectively merge indexes without knowing index contents.
3) We perform extensive experiments to evaluate the efficiency of the TBMSM scheme on a real-world dataset and achieve a logarithmic search time.

E. ROADMAP

The rest of the paper is organized as follows. First, we review the related work in section II. Then, we formulate the problem in section III and introduce the proposed scheme in section IV. Section V presents security and performance analysis. Section VI presents the performance evaluation. Finally, we conclude the paper in section VII.

II. RELATED WORK

In this section, we review two categories of related work: searchable encryption and order-preserving encryption.

A. SEARCHABLE ENCRYPTION

Searchable encryption provides a secure search service over encrypted data. Song *et al.* [2] proposed the searchable symmetric encryption(SSE) scheme that achieved ciphertext search. Goh [3] proposed a more secure SSE scheme using Bloom filter. However, a false positive may cause misjudgment [4]. Later, Curtmola *et al.* [5] proposed other schemes: SSE-1 and SSE-2. In term of efficiency, SSE-1 was better than SSE-2. In term of security, SSE-2 was safer. However, these works mostly focus on the single keyword or boolean search and don't support ranked search. Wang *et al.* [7] raised a secure ranked keyword search scheme which returned the top-k relevant files and was only designed only for single-keyword search.

The multi-keyword ranked search allows users to input multiple query keywords for personalized queries. In [9], Cao *et al.* proposed the first secure multi-keyword ranked search scheme over encrypted cloud data (MRSE), and the documents are ranked by the ''inner product'' between file vectors and query vectors. However, they do not consider the weight of different keywords. The work of [10]–[12] enriched the multi-keyword search. Wang *et al.* [13], Chuah and Hu [14] proposed multi-keyword fuzzy search scheme aimed at the tolerance of both slight typos and format inconsistencies for users' input. Zhang *et al.* [16] proposed a secure ranked multi-keyword search scheme in a multi-owner model (PRMSM) that not only allows the cloud server to perform a multi-keyword search without knowing any sensitive information, but also enable the data owner to flexibly change the encryption key. However, these schemes rarely focus on query efficiency.

Practically, query efficiency is one of the most important indicators of the user experience. Kamara and Papamanthou [17] proposed a secure search scheme based on the tree-based index, which can efficiently perform searches. However, it is designed only for a single keyword search. Later, Xu *et al.* [18], [20] presented an efficient multi-keyword ranked search scheme (MKQE) that enabled a dynamic keyword dictionary and improved the precision of the search. Sun *et al.* [19] created a privacy-preserving multi-keyword text search scheme. They divided the vector index into multiple layers and proposed a tree-based index structure by applying the MD-algorithm [21] that

realized more efficient search functionality, yet resulting in a loss of precision. Xia *et al.* [22] constructed a tree-based index structure and proposed a greedy depth-first search (GDFS) algorithm that achieved higher search efficiency. Unfortunately, these works don't consider multiple data owners scenario. Dong *et al.* [23] considered a practical scenario where multiple users share data via an untrusted third party. To implement it, the authors proposed a novel multi-user searchable data encryption scheme based on proxy cryptography. Different from the existing searchable encryption schemes, their scheme allowed the users to update the shared data set and each user can be reader and writer simultaneouly. Furthermore, the rigious proof had been represented to prove the security of their scheme. Popa *et al.* [24] focused on web applications and proposed a new platform Mylar which is a combination of system techniques and novel cryptographic primitives, including data sharing, computing over encrypted data and verifying application code. The results with 6 applications showed that Mylar is a good multi-user web application with data sharing. Cui *et al.* [25] proposed a secure and effective Near-duplicate detection (NDD) system over encrypted in-network storage which supported multi-user and multi-key searchable encryption. However, those schemes cannot solve the multi-keyword ranked search problem in the multi-user setting. Therefore, their schemes cannot directly be deployed for addressing our problem. In [26], Yao *et al.* proposed a multi-source encrypted indexes merge (MEIM) mechanism, where the cloud can merge the encrypted indexes from data owners without knowing the index content. They focused on personal health records, and only considered a numerical ''attribute value'' for each attribute while ignoring queries on data files that must be built on vectors.

B. ORDER-PRESERVING ENCRYPTION

The order-preserving encryption (OPE) is used to preserve numerical order for plaintexts [27]. Boneh *et al.* [28], [29] proposed the order-revealing encryption (ORE) schemes to achieve the best-possible security. In [30], Chenette *et al.* built the first efficiently implementable order-revealing encryption. Yao *et al.* [31] proposed a novel multiple order-preserving symmetric encryption (MOPSE) scheme to encrypt personal health record under the multi-user setting. However, these schemes cannot support additive order-preserving. In addition, Yi *et al.* [32] proposed an order-preserving function to encode data from different sensors in the sensor network. Nevertheless, it only considered range queries while ignoring the multi-keyword ranked search.

III. PROBLEM FORMULATION

A. SYSTEM MODEL

In Fig. 1, three entities, i.e., data owners, the cloud server, and data users, make up this system model. Data owners have a large collection of files F. To enable efficient multi-keyword search on the encrypted files, each data owner first builds a secure searchable tree-based index I. Then, data owners

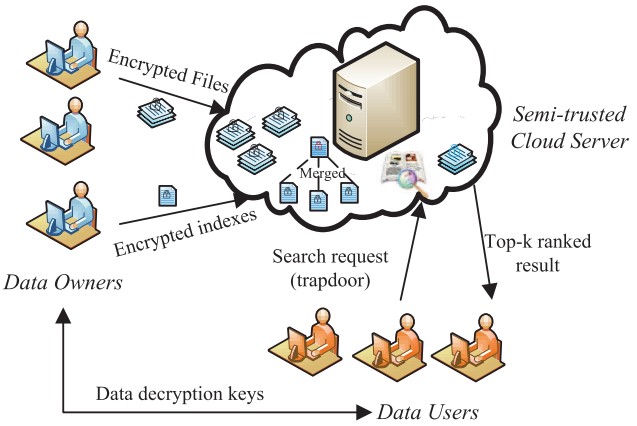


FIGURE 1. System model.

encrypt their data files F with their keys and outsource both the encrypted tree-based index and data files to the cloud server. When receiving the tree-based indexes, the cloud server merges multiple encrypted indexes without compromising data owners' privacy. When the data user searches t keywords over the encrypted files and fetch k encrypted files, he first computes the trapdoors T, and submits T and k to the cloud server. When receiving the trapdoors T and k, the cloud server begins searching the merged index tree I and returns the corresponding collection of the top-k ranked encrypted files.

B. THREAT MODEL

In the TBMSM, we assume both data owners and data users are trusted. However, we consider the cloud server to be ''curious but honest,'' which is the same as the previous works [7], [9]. This means it follows the proposed protocol, but curious about the actual information of encrypted files, the trapdoor, and relevance scores. Preserving the access pattern is too much expensive because the algorithm must access the entire file set [33]. For efficiency, we do not protect the access pattern in our scheme.

C. DESIGN GOALS

To enable an efficient ranked multi-keyword search for multiple data owners over encrypted cloud data, our scheme aims to achieve the following goals:

1) **Multi-keyword Ranked Search for Multiple Data Owners:** This scheme not only allows multi-keyword searches over encrypted cloud data (which are encrypted with different keys for different data owners) but also allow the cloud server to return the ranked top-k encrypted files.
2) **Search Efficiency:** We explore a tree-based index structure and an efficient search algorithm. The cloud server will merge encrypted indexes without knowing the corresponding sensitive information. The authenticated data user only needs to encrypt query keywords once to efficiently retrieve all files of interest.

3) **Security:** The scheme proposed should achieve the following three security goals: (1) Keyword Semantic Security (Definition 1). We will prove that TBMSM is semantically secure against the chosen keyword attack under the selective security model. (2) Keyword Security (Definition 2). We will prove that TBMSM realized keyword privacy in the random oracle model. (3) Relevance Score Security. We need to ensure that the cloud server cannot infer the actual value of the encoded relevance scores.

Definition 1: Let $\mathcal{A}$ be a probability polynomial time adversary, he can ask the challenger $\mathcal{C}$ to return the ciphertext with the corresponding keywords. The adversary $\mathcal{A}$ sends two keywords w_1 and w_2 with equal length to $\mathcal{C}$. Then the challenger $\mathcal{C}$ randomly sets $v \in \{1, 2\}$ and sends the ciphertext $\widehat{w_v}$ to the adversary $\mathcal{A}$, who continues to ask the challenger $\mathcal{C}$ for the ciphertext of keyword w, such that $w \notin \{w_1, w_2\}$. Finally, the adversary $\mathcal{A}$ outputs the guess $v' \in \{1, 2\}$. We define the probability that the adversary $\mathcal{A}$ breaks TBMSM to be $P[v' = v]$. If $P[v' = v]$ is negligible, TBMSM is semantically secure against the chosen keyword attack under the selective security model.

Definition 2: Let $\mathcal{A}$ be a probability polynomial time adversary, he can ask the challenger $\mathcal{C}$ to return the ciphertext of corresponding keywords. The challenger $\mathcal{C}$ randomly chooses a keyword w and sends the ciphertext $\widehat{w}$ to $\mathcal{A}$. Finally, the adversary $\mathcal{A}$ outputs the guess w'. We define the probability that $\mathcal{A}$ breaks the encrypted keyword to be $P[w' = w]$. If $P[w' = w]$ is negligible, TBMSM realized keyword privacy.

D. NOTATIONS

For clarity, we introduce the main notations mentioned in this paper.

- O - the collection of data owners, denoted as a set of m data owners $O = (O_1, O_2, ..., O_m)$.
- I - the collection of data owners' index, denoted as a set of m indexes $I = (I_1, I_2, ..., I_m)$.
- U - data users collection, denoted as a set of n data users $U = (U_1, U_2, ..., U_n)$.
- W - the collection of keywords, denoted as a set of η keywords $W = (w_1, w_2, ..., w_\eta)$.
- $\widehat{W_i}$ - the keyword collection encrypted by O_i, denoted as $\widehat{W_i} = (\widehat{w}_{i,1}, \widehat{w}_{i,2}, ..., \widehat{w}_{i,\eta})$.
- F_i - the plaintext file collection of O_i, denoted as a set of d files $F_i = (F_{i,1}, F_{i,2}, ..., F_{i,d})$.
- C_i - the encrypted file collection of O_i, denoted as a set of d files $C_i = (C_{i,1}, C_{i,2}, ..., C_{i,d})$.
- Q_i - a subset of W, indicating the keywords in a search request that are submitted by U_i, denoted as $Q_i = (q_{i,1}, q_{i,2}, ..., q_{i,t})$.
- T_{Q_i} - the trapdoor for Q_i, denoted as $T_{Q_i} = (T_{q_{i,1}}, T_{q_{i,2}}, ..., T_{q_{i,t}})$.

E. PRELIMINARIES

In this subsection, we introduce some necessary techniques used in the study.

1) BILINEAR PARING

Let $G_1$and G_2 denote two cyclic groups of prime order p. We view G_1 as an additive group and G_2 as a multiplicative group. A bilinear map $e : G_1 \times G_1 \rightarrow G_2$ satisfies the following properties:

- **Bilinear:** $\forall g, h \in G_1, \forall x, y \in Z_p, e(g^x, h^y) = e(g, h)^{xy}$.
- **Computable:** There is a polynomial time algorithm to compute $e(g, h) \in G_2$, for any $g, h \in G_1$.
- **Non-degenrate:** If g is a generator of G_1,the $e(g, g)$ is a generator of G_2.

2) DECISIONAL BILINEAR DIFFIE-HELLMAN (DBDH) ASSUMPTION

The DBDH problem is as follows: given $a, b, c, z \in Z_q$ as input, whether we can distinguish the tuple $(g, g^a, g^b, g^c, e(g, g)^{abc})$ from the tuple $(g, g^a, g^b, g^c, e(g, g)^z)$. The DBDH assumption states that there is no polynomial-time algorithm that has a non-negligible advantage in solving the DBDH problem.

3) VECTOR SPACE MODEL

The vector space model along with TF $\times$ IDF rule is a popular information retrieval model [34], where TF denotes the frequency of a given keyword appearing in the file and IDF is the logarithm of the total number of files divided by the number of files containing the keyword and get value obtained the logarithm. There are many variations of the TF $\times$ IDF weighting scheme. Without loss of generality, we choose a commonly used formula to calculate the relevance score of the document [35]. Given a data file set $F = \{F_1, \cdots, F_d\}$ and a keyword set $W = \{w_1, \cdots, w_\eta\}$, we compute the relevance score between F_b ($b \in [1, d]$) and an arbitrary keyword w_j ($j \in [1, \eta]$) with the following method.

$$\mathsf{Score}(F_b, w_j) = \frac{1}{|F_b|}(1 + \ln f_{F_b, w_j}) \ln (1 + \frac{N}{f_{w_j}}), \quad (1)$$

where $|F_b|$ denotes the length of the file F_b, f_{F_b,w_j} denotes the frequency of the keyword w_j in the file F_b, f_{w_j} denotes the number of files containing keyword w_j, and N denotes the total number of files, i.e., $|F|$. Here, we denote the vector of the relevance score of the keywords in the file F_b by $\mathbf{D}_b = \{\mathsf{Score}(F_b, w_1), \mathsf{Score}(F_b, w_2), \cdots, \mathsf{Score}(F_b, w_\eta)\} = \{s_{b,1}, \cdots, s_{b,\eta}\}$. Note that the length of the vector of all files is the same (i.e., η); if the file F_b does not have the keyword w_j, the corresponding jth element in D_b should be set as 0.

4) ADDITIVE ORDER AND PRIVACY PRESERVING FUNCTION (AOPPF)

Chor *et al.* [33] proposed an additive order and privacy preserving function, which helps data owners encrypt the relevance score using a different function. It allows the cloud server to accurately rank the search result files.

$$F^y_{aoppf}(x) = \sum_{0 \le j,k \le \tau} A_{j,k}.m(x, j).m(y, k) + r_{aof} \quad (2)$$

where $A_{j,k}$ and τ denote the coefficient of $m(x, j).m(y, k)$ and the degree of $F^{y}_{aoppf}(x)$, respectively. The function $m(x, \cdot)$ is used as preserving the order of relevance score x, the function $m(y, \cdot)$ is deployed for processing the data owner's ID y, and r_{aof} is the disturbing part. The function $m(x, j)$ is defined as follows: $m(x, 1) = x, m(x, 0) = 1$ and $m(x, j) = (m(x, j-1) + \alpha \cdot x) \cdot (1 + \lambda)$ if $j > 1$, where α and λ are two constant numbers. Since $\sum_{0\le j,k\le\tau} A_{j,k} \cdot (m(x+1, j) - m(x, j)) \cdot m(y, k) \ge \sum_{0\le j,k\le\tau} A_{j,k} \cdot ((1+\lambda)^{j-1} + \alpha \cdot \sum_{1\le i\le j-2}(1+\lambda))$, we let l to be an integer such that $2^{l-1} \le \sum_{0\le j,k\le\tau} A_{j,k} \cdot ((1+\lambda)^{j-1} + \alpha \cdot \sum_{1\le i\le j-2}(1+\lambda)) \le 2^{l}$, and the parameter r is an integer smaller than $l-1$. Therefore, the disturbing part r_{aof} belong to $\{0, 1, \cdots, 2^{l-1}\}$.

5) KEYWORD BALANCED BINARY TREE (KBB-TREE)

To improve the efficiency of the search, Xia *et al.* [22] first proposed the keyword balanced binary tree. However, it does not support the multiple data owners' model. In our scheme, each data owner builds a secure keyword balanced binary tree and outsource them to the cloud server. The cloud server merges those index trees and performs the efficiently multi-keyword search. Each node in the index tree stores a vector $\mathbf{D}$ whose elements are the relevance scores. We define the node in the index tree as

$$u_{node} = \langle ID, F_{ID}, \mathbf{D}, P_l, P_r \rangle \tag{3}$$

where ID, F_{ID}, and O_{ID} denote the id of node, file and data owner, respectively. P_r denotes the pointers to the right child of the u_{node}, and P_l denotes the pointers to the left child. The detailed construction of the KBB-tree and the merging method will be discussed in section IV.

IV. TREE-BASED MULTI-KEYWORD RANKED SEARCH SCHEME

In this section, we detail our TBMSM scheme in the following aspects: Overview, System Setup, Keyword Encryption, Index Construction, Secure Indexes Merge, Trapdoor Generation, Efficient Search.

A. OVERVIEW

To meet the requirements of efficient multi-keyword ranked search in multiple data owners model, we propose a novel TBMSM mechanism. Fig. 2 shows the working processes of TBMSM.

1) DATA OWNERS

(1) *KeywordsEnc* encrypts the keyword with data owners' secret key $k_{o_{i,w}}$; (2) *FilesEnc* utilizes the traditional symmetric encryption algorithm to encrypt data owners' files; (3) *IndexesEnc* builds the tree-based index for each data owner and encrypts the KBB-tree with AOPPF. (4) data owners upload encrypted keywords, files and KBB-trees to the cloud server.

2) DATA USERS

(1) *TrapdoorGen* generates trapdoors with data users' secret key $k_{u_{i,w}}$, and then submits trapdoors and the number of extracting files k to the cloud server; (2) *FilesDec* decrypts encrypted files.

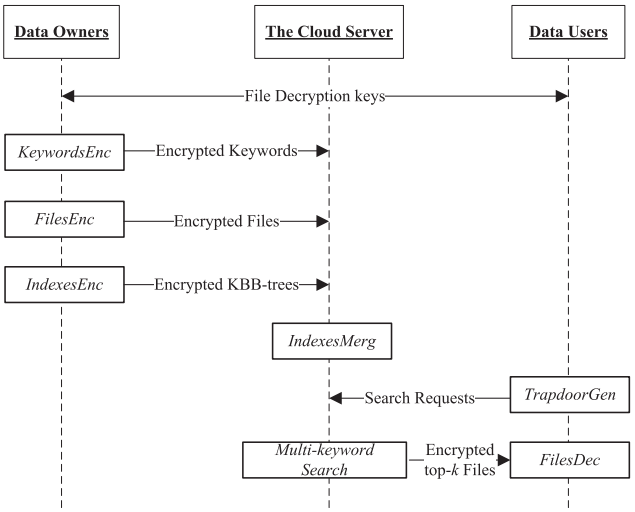


FIGURE 2. TBMSM working processes.

3) THE CLOUD SERVER

(1) *IndexesMerg* merges multiple encrypted KBB-trees; (2) *Multi-keyword Search* runs the *DFS* algorithm to find out the corresponding files and returns the corrected top-k encrypted files to data users.

B. SYSTEM SETUP

Let k denote the security parameter. Both g and g_1 are the generators of cyclic groups G_1 and G_2 with prime order p. Our algorithm randomly produces $k_{u_j,w} \in Z_p^+$, $k_{o_i,f} \in Z_p^+$, $k_{o_i,w} \in Z_p^+ \leftarrow (0, 1)^*$, where $k_{u_j,w}$ is the private key of data user U_j used to generate the trapdoor, $k_{o_i,f}$ and $k_{o_i,w}$ are the secret keys of data owner O_i for encrypting files and keywords. In addition, let $H(\bullet)$ denote a public hash function whose output belongs to Z_p^+.

C. KEYWORD ENCRYPTION

To meet the security search requirements, keyword encryption should satisfy the following conditions. First, different data owners can encrypt the same keyword with different private keys, and don't need to share the keys with others. Second, the same keyword should be encrypted to different ciphertext at different times.

We assume that data owner O_i wants to encrypt the ath keyword $w_{i,a}$

$$\widehat{w}_{i,a} = (g^{k_{o_i,w} \cdot H(w_{i,a}) \cdot r_o}, g^{k_{o_i,w} \cdot r_o}) \tag{4}$$

where r_o is a random number. For clarity, we let $EK_1 = g^{k_{o_i,w} \cdot H(w_{i,a}) \cdot r_o}$ and $EK_2 = g^{k_{o_i,w} \cdot r_o}$, so

$$\widehat{w}_{i,a} = (EK_1, EK_2) \tag{5}$$

D. INDEX CONSTRUCTION

Each data owner builds a KBB-tree for his/her data files. Given a data file set $F = \{F_1, \cdots, F_d\}$, data owner O_i

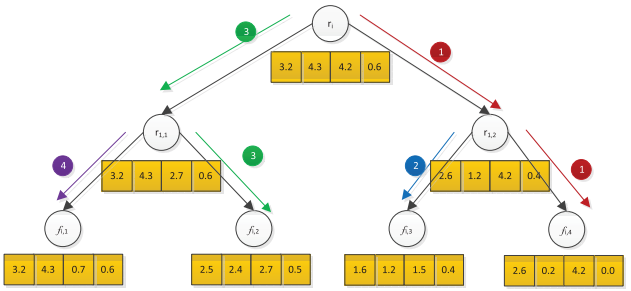


FIGURE 3. Illustration of the DFS-algorithm on the KBB-tree.

first utilizes the identical method in Sect. III-E.3 to calculate the vector space $\mathbf{D}_b$ for each file F_b, where $b \in [1, d]$. To guarantee the security of the vector space $\mathbf{D}_b$, data owner O_i adopts the AOPPF $F_{aoppf}^{H(ID_i)}(\bullet)$ to encode each element in D_b:

$$V_{i,b,j} = F_{aoppf}^{H(ID_i)}(S_{i,b,j}) \tag{6}$$

where $H(\bullet)$ denotes a public hash function, $V_{i,b,j}$ is the encoded data of $S_{i,b,j}$. The vector $\mathbf{D}_b$ will be encoded to $\widehat{\mathbf{D}}_b$, where the jth element $\widehat{\mathbf{D}}_b(j)$ is equivalent to $V_{i,b,j}$. Besides, data owner O_i takes $S_{i,b,j}$ as the input of Eq. 7 to calculate the function $V_{b,j}^i$, and denotes the function vector as $\widehat{\mathbf{V}}^b = \{V_{b,1}^i, \cdots, V_{b,j}^i\}$ $(b \in [1, d], j \in [1, \eta])$.

$$V_{b,j}^i = F_{aoppf}^i(S_{i,b,j}) \tag{7}$$

To build a KBB-tree for all data files, data owner O_i generates a node for each data file F_b as $u_{i,b} = \langle ID_i, F_b, \widehat{\mathbf{D}}_b, \widehat{\mathbf{V}}^b, P_b^l, P_b^r \rangle$. Subsequently, the Algorithm 1 takes all nodes of each owner ($\mathbf{linkedList}_i = \{u_{i,1}, \cdots, u_{i,d}\}$) as inputs,[1] and outputs the root of the KBB tree.

Figure 3 takes four files ($F_i = \{f_{i,1}, f_{i,2}, f_{i,3}, f_{i,4}\}$) as an example to show how to construct the KBB tree.

E. MULTI-SOURCE INDEXES MERGE

When having received the index tree set $I = \{I_1, I_2, \cdots, I_m\}$ from different data owners, the cloud server merges them into one index tree I_{merged} according to the following steps.

First, the cloud server initializes an empty **linkedList** and inserts $u_{i,root}$ ($u_{i,root}$ denotes the root of I_i, $i \in [1, m]$) to **linkedList**. Second, the cloud server takes the **linkedList** as inputs of the Algorithm 1 and gets the root of merged tree.

As data owners utilize different encoding functions to encode their relevance scores, the cloud server therefore cannot directly compare the size of the encoded scores from different data owners. In what follows, we take two files (F_1 and F_2) and the h-th keyword w_h as an example to explain how to compare the size of the encoded scores with different encoding functions.

Assume the F_1 of data owner O_1 and F_2 of data owner O_2 contain the hth keyword w_h. The relevance score $S_{1,1,h}$ and $S_{2,2,h}$ are encoded to $V_{1,1,h}$ and $V_{2,2,h}$. First, the cloud server chooses the function $V_{2,h}^i = F_{aoppf}^i(S_{2,2,h})$ (The cloud server can also choose $V_{1,h}^i = F_{aoppf}^i(S_{1,1,h})$) and substitutes ID_1 for the variable i and get $V_{2,h}^{ID_1}$. If $V_{1,1,h} > V_{2,h}^{ID_1}$, the relevance score of F_1 is bigger than F_2, otherwise, the relevance score of F_2 is bigger than F_1.

[1]Note that if the number of nodes is not power of two, some dummy nodes (e.g., $u =$ **NULL**) need be introduced for constructing the KBB tree.

Algorithm 1 BuildIndex

```
Input: linkedList
Output: Encrypted index tree I
if linkedList.size() == 1 then
    return the element of linkedList;
else
    Initialize an empty list tempNodeList;
    i ← 1;
    for each pair ⟨u_i, u_{i+1}⟩ in linkedList do
        Initialize an empty parent node u;
        u.ID ← ID_{i,i+1};
        u.F ← NULL;
        u.P^l_{i,i+1} ← u_i;
        u.P^r_{i,i+1} ← u_{i+1};
        for j in [1, ··· , η] do
            if D̂_i(j) > D̂_{i+1}(j) then
                u.D̂_{i,i+1}(j) ← D̂_i(j);
                u.V̂^{i,i+1}(j) ← V̂^i(j);
            else
                u.D̂_{i,i+1}(j) ← D̂_{i+1}(j);
                u.V̂^{i,i+1}(j) ← V̂^{i+1}(j);
            end
        end
        Insert u to tempNodeList;
        i ← i + 2;
    end
    Replace linkedList with tempNodeList;
    BuildIndex(linkedList)
end
```

F. TRAPDOOR GENERATION

In the multi-owner model, it is not advisable for the data user to communicate with each data owner to generate the trapdoor, so we should satisfy the following two conditions. First, the data user can generate the trapdoor without communicating with others. Second, the same search keyword should generate different trapdoors at different times.

We assume data user U_i wants to search keyword w_t, and then computes the trapdoor as

$$T_{w_t} = (g^{k_{u_i,w} \cdot H(w_t) \cdot r_u}, g^{k_{u_i,w} \cdot r_u}) \tag{8}$$

where r_u is a random number. For clarity, we let $T_1 = g^{k_{u_i,w} \cdot H(w_t) \cdot r_u}$ and $T_2 = g^{k_{u_i,w} \cdot r_u}$, so $T_{w_t} = (T_1, T_2)$ and submits it to the cloud server.

G. EFFICIENT SEARCH

When the cloud server receives a search request, it first converts the trapdoor into a search vector and then calls the DFS-algorithm described in algorithm 2 and returns the top-k

related files. We assume the cloud server received a trapdoor T_{w_t}, and for every encrypted keyword in the keyword set $\widehat{W}_i$, it first computes

$$\begin{aligned} e(EK_1, T_2) &= e(g^{k_{o_i,w} \cdot H(w_{i,a}) \cdot r_o}, g^{k_{u_i,w} \cdot r_u}) \\ &= e(g, g)^{k_{o_i,w} \cdot H(w_{i,a}) \cdot r_o \cdot k_{u_i,w} \cdot r_u} \end{aligned}$$

The cloud can judge whether w_t belongs to the keyword set if the following equation is true and set the ath dimension of the query vector to 1:

$$\begin{aligned} e(EK_2, T_1) &= e(g^{k_{o_i,w} \cdot r_o}, g^{k_{u_i,w} \cdot H(w_t) \cdot r_u}) \\ &= e(g, g)^{k_{o_i,w} \cdot r_o \cdot k_{u_i,w} \cdot H(w_t) \cdot r_u} \\ &= e(EK_1, T_2) \end{aligned}$$

Algorithm 2 DFS

Input: The IndexTree's node u, query vector q and returned file number k
Output: The list *RList* of top-k ranked encrypted files
if *the node u is not a leaf node* **then**
  if *getRScore*$(u, q) >$ *minScore* **then**
    DFS(*u.getLeftNode*(), q, k);
    DFS(*u.getRightNode*(), q, k);
  end
else
  if *getRScore*$(u, q) >$ *minScore* **then**
    if *RList.size*() == k **then**
      Delete the element with the smallest encoded score from *RList*;
    end
    Insert a new element and sort the element of *RList*;
    Set *minSocre* to be equal to the smallest encoded score in *RList*;
  end
end

Fig. 3 shows an example of a search process with query vector $q = (0, 0, 1, 1)$, and the parameter $k = 3$. The search starts at the root node, and reaches the first leaf node $f_{i,4}$ with the query relevance score 4.2. Then, the leaf node $f_{i,3}$ and $f_{i,2}$ are reached with relevance scores 1.9 and 4.5 respectively. Next, the DFS-algorithm will reach the leaf node $f_{i,1}$ with the relevance score 1.3, which is smaller than the smallest score in *RList*. Finally, the elements of *RList* are sorted by relevance score.

V. SECURITY AND PERFORMANCE ANALYSIS

A. SECURITY ANALYSIS

Data files are encrypted by a symmetric encryption algorithm before uploading. As long as the algorithm is safe, the cloud server will not know the contents of files.

1) KEYWORDS AND TRAPDOORS

We give the security analysis of keywords and trapdoors according to the following two theorems.

Theorem 1: Based on the DBDH assumption, TBMSM is semantically secure against the chosen keyword attack under the selective security model.

Proof: We first consider a game played between adversary $\mathcal{A}$ and challenger $\mathcal{C}$. We assume adversary $\mathcal{A}$ has a non-negligible advantage ϵ as the attacker in this game. In the initialization phase, challenger $\mathcal{C}$ sets μ randomly. If $\mu = 0, \mathcal{C}$ sends $(A, B, C, Z) = (g^a, g^b, g^c, g^{abc})$ to $\mathcal{A}$; If $\mu = 1, \mathcal{C}$ sends $(A, B, C, Z) = (g^a, g^b, g^c, g^z)$ to $\mathcal{A}$, where $a, b, c, z \in Z_p$ are randomly generated.

Setup: Challenger $\mathcal{C}$ sends public keys (g, g^a, g^b, g^c, Z) to the adversary $\mathcal{A}$.

Phase 1: $\mathcal{C}$ initializes a set of keywords W and sets it to an empty value. $\mathcal{A}$ can choose any keyword w and ask $\mathcal{C}$ to generate corresponding keyword ciphertext $\widehat{w}$. If $w \in W$, $\mathcal{C}$ just sends $\widehat{w}$ to $\mathcal{A}$; Otherwise, $\mathcal{C}$ adds w to W and sends $\widehat{w}$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ sends two keywords w_1 and w_2 with equal length to $\mathcal{C}$, where $w_1, w_2 \notin W$, $\mathcal{C}$ randomly sets $v \in \{1, 2\}$ and sends the ciphertext $\widehat{w_v} = (Z^{H(w_v)}, Z)$ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ repeats phase 1 and the only limit is $w \notin \{w_1, w_2\}$.

Guess: $\mathcal{A}$ outputs the guess $v' \in \{1, 2\}$. If $v' = v$, then $\widehat{w_v}$ is the ciphertext of w_v, and $\mathcal{A}$ outputs $\mu = 0$; Otherwise it outputs $\mu = 1$.

Evidently, $\mathcal{A}$ will output $v' = v$ with probability $1/2 + \epsilon$ and output $v' \neq v$ with probability $1/2$. Therefore, $\mathcal{A}$ can win the game with probability $1/2+\epsilon/2$. In other words, TBMSM is secure against the chosen keyword attack. □

Theorem 2: Based on the discrete logarithm assumption, TBMSM realized keyword privacy in the random oracle model.

Proof:

We consider a game played between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ as follows.

Setup: The challenger $\mathcal{C}$ sends public keys (g, g^{k_1}, g^{k_2}) to the adversary $\mathcal{A}$.

Phase 1: $\mathcal{C}$ initializes a set of keywords W and sets it to an empty value. $\mathcal{A}$ can choose any keyword w and ask $\mathcal{C}$ to generate corresponding keyword ciphertext $\widehat{w}$. If $w \in W$, $\mathcal{C}$ just sends $\widehat{w}$ to $\mathcal{A}$; Otherwise, $\mathcal{C}$ adds w to W and sends $\widehat{w}$ to $\mathcal{A}$.

Challenge: After $\mathcal{A}$ has submitted t search requests, $\mathcal{C}$ randomly chooses keyword w' and returns encrypted keyword $\widehat{w'} = (g^{k_1 \cdot H(w') \cdot r_1}, g^{k_1 \cdot r_1})$ and trapdoor $T_{w'} = (g^{k_2 \cdot H(w') \cdot r_2}, g^{k_2 \cdot r_2})$ to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs the guess w'' for w' and sends it to $\mathcal{C}$. $\mathcal{C}$ returns the encrypted keyword $\widehat{w''}$ to $\mathcal{A}$. If $T_{w'}$ matches $\widehat{w''}$, then $\mathcal{A}$ wins the game.

Before outputting the guess w', $\mathcal{A}$ already has t keywords and their encrypted keywords. Therefore, the remaining keyword set is $\eta - t$, where η is the size of keywords. In addition, because the discrete logarithm problem is hard in polynomial time, the probability that $\mathcal{A}$ guess the correct w' from $\widehat{w'}$ or $T_{w'}$ is a negligible probability ϵ. Therefore, $\mathcal{A}$ can win the game with probability $\frac{1}{\eta - t} + \epsilon$. □

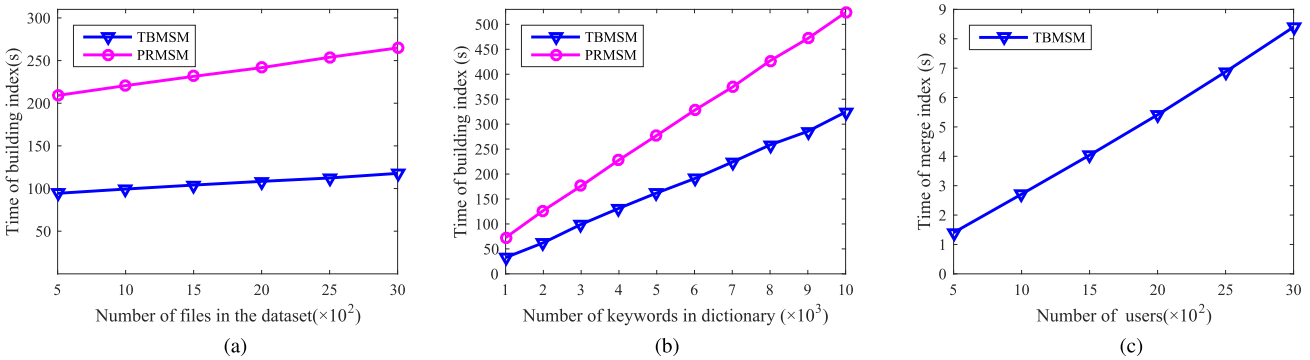


FIGURE 4. Time cost of index construction. (a) keyword dictionary u = 4000. (b) data set d = 1000. (c) keyword dictionary u = 4000.

2) RELEVANCE SCORES

We encode relevance scores with the *AOPPF* proposed in [33]. We assume the data owner inputs score s and ID_i, and sends the encoded score $F_{aoppf}^{H(ID_i)}(s)$ to the cloud server. When the cloud server collects m encoded scores for the same score s, it can construct m equations. However, there are $m+1$ unknown variables, so the *AOPPF* cannot be broken based on those information. Therefore, the privacy of the relevance scores has also been protected.

B. PERFORMANCE ANALYSIS

1) INDEX ENCRYPTION

Assume that the keyword dictionary and file numbers for each index are defined as u and d. $\mathcal{O}(d)$ nodes are generated during the index tree construction, and for each node it takes $\mathcal{O}(u)$ time to encrypt it. The time complexity of building an index is $\mathcal{O}(ud)$.

2) INDEX MERGE

Assume that there are m index trees to merge. Merging m indexes is equivalent to merging their roots. It takes $\mathcal{O}(um)$ to merge these encrypted indexes. Therefore, the time complexity of the merging index is $\mathcal{O}(um)$.

3) QUERY

We assume that the number of leaf nodes containing at least one query keyword is n. The time cost to calculate the relevance score is $\mathcal{O}(u)$, and the height of the merged index is $\log d + \log m$. The search time complexity is less than $\mathcal{O}(un(\log d + \log m))$, as the top-k queries of n are larger than the number of required files k.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the efficiency of the TBMSM. First, we introduce the evaluation settings, then compare our proposed scheme with the PRMSM.

A. EVALUATION SETTINGS

Our experiments are conducted using the Java programming language on a PC with a 3.3 GHz Intel Core CPU and 8 GB memory. We use the real datasets, the Internet Request For Comments (RFC) [36] for simulation experiments. This dataset contains 7875 plain text files with a total size of approximately 393MB. Then we extract keywords from the RFC files.

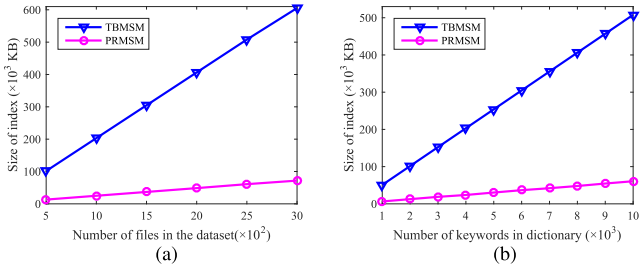


FIGURE 5. Index size. (a) keyword dictionary u = 4000. (b) data set d = 1000.

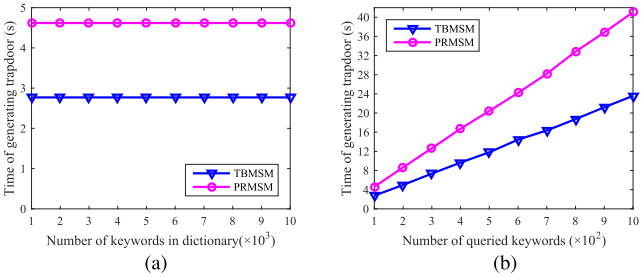


FIGURE 6. Time cost of generating trapdoors. (a) queried keywords t = 100. (b) keyword dictionary u = 4000.

B. EVALUATION RESULTS

1) INDEX CONSTRUCTION

Fig. 4 shows the time cost of index construction. Fig. 4a shows the time cost of index construction for the same size of keyword dictionary, which is almost linear with the number of files. This is consistent with our performance analysis. The TBMSM consumes much less time than PRMSM. Fig. 4b shows that, given the same number of files ($d = 1000$), TBMSM grew by an average of 29.16s, while PRMSM grew by an average of 45.12s. Fig. 4c shows the time cost of

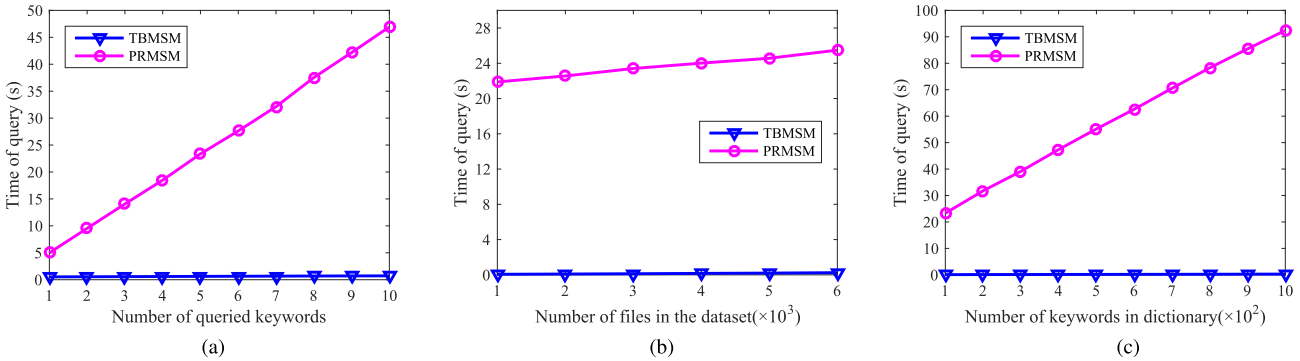


FIGURE 7. Time cost of search. (a) data set d = 2000 and keyword dictionary u = 100. (b) queried keywords t = 5 and keyword dictionary u = 100. (c) queried keywords t = 5 and date set d = 2000.

the merging operation with the same keyword dictionary ($u = 4000$) and grows linearly with the number of users. When the number of data owners belongs to {500, 3000}, the time cost values within {1.403s, 8.399s}, which is acceptable. On the other hand, as shown in Fig. 5 (due to the use of balanced binary tree), the TBMSM needs more storage space for the index. However, this is not a problem for the cloud platform.

2) TRAPDOOR GENERATION

Fig. 6a shows that both the TBMSM and PRMSM are unaffected by the keyword dictionary size for the same number of queried keywords. The TBMSM requires approximately 2.769s to generate trapdoors, while the PRMSM requires 4.618s. In Fig. 6b, given the same number of keyword dictionary sizes ($u = 4000$), the trapdoor generation time for the TBMSM ranges from 2.769s to 23.588s, while it ranges from 4.618s to 41.044s for PRMSM.

3) SEARCH

Fig. 7 shows that our proposed scheme consumes less time search compared to PRMSM, which matches our search analysis. Fig. 7a shows that given d = 2000 and u = 100, when the number of queried keywords belongs to {1, 10}, the search time in the PRMSM ranges from 4.992s to 46.932s, while our method's search time ranges from 0.572s to 0.72s. Fig. 7b shows that the number of files in the dataset has little effect on the search. Fig. 7c shows that given the same size of queried keywords (t = 5), and the same dataset (d = 2000), when the size of the keyword dictionary increases from 100 to 1000, the TBMSM's search time ranges from 0.63s to 0.834s, only an average of 0.02s, while the PRMSM ranges from 23.31s to 92.947s.

VII. CONCLUSION

In this study, we consider a multiple data owners model in cloud computing and propose an efficient ranked multi-keyword search scheme over encrypted data. First, we propose a novel secure search protocol that allows different data owners to encrypt the files and indexes with different keys. Then, we construct a tree-based index structure for each data owner and encrypt with AOPPF. Meanwhile, the TBMSM allows the cloud server to merge encrypted indexes without knowing any information. The experiment results obtained using the RFC dataset demonstrate that the TBMSM is an efficient mechanism.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *Nat. Inst. Standard Technol.*, vol. 53, no. 6, p. 50, 2011.
[2] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2000, pp. 44–55.
[3] E. Goh, "Secure indexes," IACR Cryptol. ePrint Arch., Tech. Rep. 2003/216, 2003.
[4] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, 2004.
[5] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, Jan. 2011.
[6] Q. Liu, G. Wang, and J. Wu, "Secure and privacy preserving keyword searching for cloud storage services," *J. Netw. Comput. Appl.*, vol. 35, no. 3, pp. 927–933, 2012.
[7] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst.*, Genova, Italy, Jun. 2010, pp. 253–262.
[8] 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, pp. 3–14, May 2017.
[9] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. INFOCOM*, Shanghai, China, Apr. 2011, pp. 829–837.
[10] A. Ibrahim, H. Jin, A. A. Yassin, and D. Zou, "Secure rank-ordered search of multi-keyword trapdoor over encrypted cloud data," in *Proc. APSCC*, Guilin, China, Dec. 2012, pp. 263–270.
[11] C. Orencik, M. Kantarcioglu, and E. Savas, "A practical and secure multi-keyword search method over encrypted cloud data," in *Proc. CLOUD*, Santa Clara, CA, USA, Jun./Jul. 2013, pp. 390–397.
[12] Z. Shen, J. Shu, and W. Xue, "Preferred keyword search over encrypted data in cloud computing," in *Proc. IWQoS*, Montreal, QC, Canada, Jun. 2013, pp. 1–6.
[13] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. INFOCOM*, Toronto, ON, Canada, Apr./May 2014, pp. 2112–2120.
[14] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," in *Proc. ICDCSW*, Minneapolis, MN, USA, Jun. 2011, pp. 273–281.
[15] S. K. Pasupuleti, S. Ramalingam, and R. Buyya, "An efficient and secure privacy-preserving approach for outsourced data of resource constrained mobile devices in cloud computing," *J. Netw. Comput. Appl.*, vol. 64, pp. 12–22, Apr. 2016.

[16] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou, "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1566–1577, May 2016.

[17] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptograph. Data Secur.*, Okinawa, Japan, 2013, pp. 258–274.

[18] Z. Xu, W. Kang, R. Li, K. Yow, and C.-Z. Xu, "Efficient multi-keyword ranked query on encrypted data in the cloud," in *Proc. ICPADS*, Singapore, Dec. 2012, pp. 244–251.

[19] W. Sun *et al.*, "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 3025–3035, Nov. 2014.

[20] R. Li, Z. Xu, W. Kang, K. C. Yow, and C.-Z. Xu, "Efficient multi-keyword ranked query over encrypted data in cloud computing," *Future Generat. Comput. Syst.*, vol. 30, pp. 179–190, Jan. 2014.

[21] M. Ondreička and J. Pokorný, "Extending Fagin's algorithm for more users based on multidimensional B-tree," in *Proc. East Eur. Conf. Adv. Databases Inf. Syst.*, Pori, Finland, 2008, pp. 199–214.

[22] 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.

[23] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," *J. Comput. Secur.*, vol. 19, no. 3, pp. 367–397, 2011.

[24] R. A. Popa *et al.*, "Building Web applications on top of encrypted data using Mylar," in *Proc. NSDI*, Seattle, WA, USA, 2014, pp. 157–172.

[25] H. Cui, X. Yuan, Y. Zheng, and C. Wang, "Enabling secure and effective near-duplicate detection over encrypted in-network storage," in *Proc. INFOCOM*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[26] X. Yao, Y. Lin, Q. Liu, and S. Long, "Efficient and privacy-preserving search in multi-source personal health record clouds," in *Proc. ISCC*, Larnaca, Cyprus, Jul. 2015, pp. 803–808.

[27] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order-preserving encryption for numeric data," in *Proc. SIGMOD*, Paris, France, 2004, pp. 563–574.

[28] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman, "Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation," in *Proc. Adv. Cryptol. (EUROCRYPT)*, Sofia, Bulgaria, 2015.

[29] K. Lewi and D. J. Wu, "Order-revealing encryption: New constructions, applications, and lower bounds," in *Proc. CCS*, Hofburg, Austria, 2016, pp. 1167–1178.

[30] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu, "Practical order-revealing encryption with limited leakage," in *Proc. FSE*, Seattle, WA, USA, 2016, pp. 474–493.

[31] X. Yao, Y. Lin, Q. Liu, and J. Zhang, "Privacy-preserving search over encrypted personal health record in multi-source cloud," *IEEE Access*, vol. 6, pp. 3809–3823, 2018.

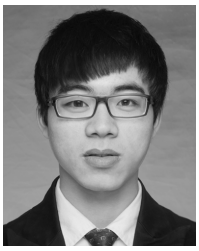
[32] Y. Yi, R. Li, F. Chen, A. X. Liu, and Y. Lin, "A digital watermarking approach to secure and precise range query processing in sensor networks," in *Proc. INFOCOM*, Turin, Italy, Apr. 2013, pp. 1950–1958.

[33] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. 36th Annu. Symp. Found. Comput. Sci.*, Oct. 1995, pp. 41–50.

[34] R. Larson, "Introduction to information retrieval," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 61, no. 4, pp. 852–853, 2010.

[35] I. H. Witten, A. Moffat, and T. C. Bell, "Managing gigabytes: Compressing and indexing documents and images," *IEEE Trans. Inf. Theory*, vol. 41, no. 6, p. 2101, Nov. 1995.

[36] *Request for Comments Database*. Accessed: 2017. [Online]. Available: https://www.ietf.org/rfc.html



TIANYUE PENG received the B.S. degree in software engineering from Hunan University, China, in 2015, where he is currently pursuing the M.S. degree in software engineering. His research interests include security and privacy issues in cloud and big data.

YAPING LIN received the B.S. degree in computer application from Hunan University, China, in 1982, the M.S. degree in computer application from the National University of Defense Technology, China, in 1985, and the Ph.D. degree in control theory and application from Hunan University in 2000. He has been a Professor and a Ph.D. Supervisor with Hunan University since 1996. From 2004 to 2005, he was a Visiting Researcher with the University of Texas at Arlington. His research interests include machine learning, network security, and wireless sensor networks.

XIN YAO received the B.S. degree in computer science from Xidian University, China, in 2011, and the M.S. degree in software engineering from Hunan University, China, in 2013, where he is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering. He is also a Visiting Student with Arizona State University, Tempe, AZ, USA. His research interests include security and privacy issues in social network, cloud, and big data.

WEI ZHANG received the B.S. degree in computer science from Hunan University, China, in 2011, where he is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering. Since 2014, he has been a Visiting Student with the Department of Computer and Information Sciences, Temple University. His research interests include cloud computing, network security, and data mining.