IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 5, 2019, accepted December 23, 2019, date of publication December 31, 2019, date of current version January 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2963096

A Privacy-Preserving Multi-Keyword Ranked Search Over Encrypted Data in Hybrid Clouds

HUA DAI[1,2], YAN JI[1], GENG YANG[1,2], HAIPING HUANG[1], AND XUN YI[3]

[1]Nanjing University of Post and Telecommunication, Nanjing 210023, China
[2]Jiangsu Security and Intelligent Processing Lab of Big Data, Nanjing 210023, China
[3]Royal Melbourne Institute of Technology University, Melbourne, VIC 3001, Australia

Corresponding author: Hua Dai (daihua@njupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872197, Grant 61572263, Grant 61902199, Grant 61972209, Grant 61672297, and Grant 61872193, in part by the Postdoctoral Science Foundation of China under Grant 2019M651919r2, in part by the Natural Science Foundation of Anhui Province under Grant 1608085MF127, in part by the University Natural Science Foundation of Anhui Province under Grant KJ2017A419, and in part by the Natural Research Foundation of the Nanjing University of Posts and Telecommunications under Grant NY217119 and Grant NY219142.

ABSTRACT With the rapid development of cloud computing services, more and more individuals and enterprises prefer to outsource their data or computing to clouds. In order to preserve data privacy, the data should be encrypted before outsourcing and it is a challenge to perform searches over encrypted data. In this paper, we propose a privacy-preserving multi-keyword ranked search scheme over encrypted data in hybrid clouds, which is denoted as MRSE-HC. The keyword dictionary of documents is clustered into balanced partitions by a bisecting k-means clustering based keyword partition algorithm. According to the partitions, the keyword partition based bit vectors are adopted for documents and queries which are utilized as the index of searches. The private cloud filters out the candidate documents by the keyword partition based bit vectors, and then the public cloud uses the trapdoor to determine the result in the candidates. On the basis of the MRSE-HC scheme, an enhancement scheme EMRSE-HC is proposed, which adds complete binary pruning tree to further improve search efficiency. The security analysis and performance evaluation show that MRSE-HC and EMRSE-HC are privacy-preserving multi-keyword ranked search schemes for hybrid clouds and outperforms the existing scheme FMRS in terms of search efficiency.

INDEX TERMS Hybrid cloud, multi-keyword ranked search, privacy-preserving, searchable encryption.

I. INTRODUCTION

Nowadays, the cloud computing technology is considered as a rapid developing and popular model of distributed computing and storage, which has the advantages of high-quality data storage, quick and convenient computing and ''on-demand service'', etc. Outsourcing service built on the cloud can effectively reduce the maintenance cost of enterprises purchasing hardware and software and managing data. Attracted by the convenience, economy and high scalability appealing features, more and more individuals and enterprises are motivated to outsource their data or computing to the cloud. However, in the outsourcing cloud, Data Owner(DO) is unable to directly control and manage data stored on the Cloud Server(CS), thus, DO cannot certain data whether be protected and whether be legally and reasonably used and computed, which leads to the privacy of data is seriously threatened. At present, the privacy protection exists in the outsourcing cloud has become a major obstacle impeding its further development [2].

In the outsourced cloud, a native scheme proposed to protect data confidentiality is to encrypt data before outsourcing data to cloud. However, encrypted data cannot be directly searched and used. When the scale of data is smaller, DO can download all data to local computer and then decrypt these data, thereby obtain needed information from plaintext data. But in the current increasingly popular Big Data applications, utilize this method will cause a huge cost of time and bandwidth in terms of acquiring needed information, therefore this method does not possess essential practicality. Therefore, it is a challenge to perform privacy-preserving ranked search over encrypted cloud data.

In this paper, we propose a privacy-preserving multi-keyword ranked search over encrypted data in hybrid clouds. The keyword partition vector model is presented, in which the keywords of documents are clustered by a given bisecting

The associate editor coordinating the review of this manuscript and approving it for publication was Luis Javier Garcia Villalba.

k-means clustering algorithm and multiple balanced partitions are generated. Keywords are with high relevance scores in a partition. And the relevance score is calculated by the Normalized Google-Distance [3]. According to the generated partitions, the document filtering bit vector (DFB-vectors) and the query filtering bit vector (QFB-vector) are defined for documents and queries respectively. The former is the index for performing efficient searches while the later is used as the query command. There are two mainly stages in the proposed scheme which are the setup stage and the search stage. In the setup stage, the keyword partitions are first clustered and then DFB-vectors are created and deployed in Pri-Cloud. Documents and the corresponding vectors are encrypted and outsourced to Pub-Cloud. In the search stage, when a query having multi-keywords is started, the corresponding QFB-vector and trapdoor are respectively generated and submitted to Pri-Cloud and Pub-Cloud. In Pri-Cloud, the QFB-vector and DFB-vectors are used to filter out the candidate documents corresponding to the target partitions of the query, and then the IDs of the candidate documents are given to Pub-Cloud. After that, Pub-Cloud use the trapdoor and candidate IDs are used to determine the result encrypted documents. Because keywords in a partition are in high relevance with each other and the keywords of a query are usually relevant to each other in practice, the number of target partitions is small and the candidate documents are less correspondingly. Therefore, the proposed scheme is efficient which is indicated in the performance evaluation result. To improve the search efficiency, an enhanced scheme EMRSE-HC is proposed, which introduces a complete binary tree-based index structure and an optimized filtering algorithm.

The contributions of this paper are as follows.

(1) We proposed a keyword partition vector model. In this model, a bisecting k-means clustering based keyword partition algorithm is proposed, which generates the balanced keyword partitions and the keyword partition based bit vectors (DFB-vector and QFB-vector). DFB-vectors are the index for searches.

(2) On the basis of the keyword partition vector model and the complete binary tree structure, we propose an efficient ranked search scheme over encrypted data in hybrid clouds. The private cloud filters out the candidate documents, and then the public cloud determines the result.

(3) We analyze the security of the proposed scheme and evaluate its search performance. The result shows that the proposed scheme is a privacy-preserving multi-keyword ranked search scheme for hybrid clouds and outperforms the existing scheme FMRS in terms of search efficiency.

II. RELATED WORK

To support the multi-keyword search over the outsourced encrypted cloud data, researchers have proposed many Searchable Encryption (SE) schemes [4]–[18].

Song *et al.* [4] proposed the first symmetric searchable encryption (SSE) scheme. Cao *et al.* [5], [6] proposed the first multi-keyword ranked search scheme. The vector space model (VSM) [19] and secure KNN [20] are adopted to achieve the privacy-preserving ranked searches. Xu *et al.* [7] proposed a two-step-ranking search scheme over encrypted cloud data which adopts the order-preserving encryption (OPE) [21], [22]. Yang *et al.* [8] proposed a fast privacy-preserving multi-keyword search scheme. It supports dynamic updates on documents. Li *et al.* [9], [10] proposed a fine-grained multi-keyword search scheme over encrypted cloud data. However, only boolean queries are supported. Xia *et al.* [11] proposed a secure and dynamic multi-keyword ranked search scheme by adopting a balanced binary tree index. Chen *et al.* [13] and Zhu *et al.* [12] proposed two different privacy-preserving ranked search schemes, which both utilize clustering algorithm to improve search efficiency.

Fu *et al.* [15] and Wang *et al.* [14] proposed multi-keyword fuzzy search schemes over encrypted outsourced data. To achieve fuzzy search, the locality-sensitive hashing functions [23], wordnet and secure KNN are adopted. Wang *et al.* [16] presented a multi-keyword fuzzy search scheme which supports range queries by adopting the locality-sensitive hashing functions, bloom filtering [24] and order-preserving encryption. Fu *et al.* [17] proposed a synonym expansion of document keywords and realized the synonym-based multi-keyword ranked search scheme. Xia *et al.* [18] proposed a multi-keyword semantic ranked search scheme where the inverted index for documents and the semantic relationship library for keywords are adopted. Fu *et al.* [25] proposed a different semantic-aware ranked search scheme which adopts the concept hierarchy and the semantic relationship between concepts.

According to the state-of-art, most of the existing works focus on the public clouds. Only Yang *et al.* [26] proposed a search scheme for the hybrid clouds which consist of the public cloud (Pub-Cloud) and the private cloud (Pri-Cloud). In the scheme, Pri-Cloud is assumed trust while Pub-Cloud is assumed honest-but-curious. Keywords of documents are equally divided into multiple partitions and document index vectors are created for each document according to the partitions. Pri-Cloud utilizes document index vectors to obtain candidate document identities and then Pub-Cloud determines the result encrypted documents whose identities are the candidates. The more partitions the searched keywords cover, the more candidate document identities Pri-Cloud obtains. The search efficiency is proportional to the number of partitions covering the queried keywords. In practical, the searched keywords are usually relevant to each other. For example, basketball, NBA, slam dunk could be the queried keywords for retrieving the interested news, and they are obviously relevant. Therefore, if the keywords with high relevance are gathered in fewer partitions, then the search efficiency will be improved when the searched keywords are relevant.

III. NOTATIONS AND PRELIMINARIES

A. NOTATIONS

- d_i — A plaintext document.
- D — A plaintext document collection, $D = \{d_1, d_2, , d_m\}$.
- V_{d_i} — The n-dimensional document vector of d_i.
- V_D — The set of document vectors of documents in D, $V_D = \{V_{d_1}, V_{d_2}, \ldots, V_{d_m}\}$.
- $\widetilde{d}_i$ — The encrypted document of d_i.
- $\widetilde{D}$ — The encrypted document collection of D, $\widetilde{D} = \{\widetilde{d}_1, \widetilde{d}_2, \ldots, \widetilde{d}_m\}$.
- $\widetilde{V}_{d_i}$ — The encrypted n-dimensional document vector of d_i.
- $\widetilde{V}_D$ — The set of encrypted documents vectors, $\widetilde{V}_D = \{\widetilde{V}_{d_1}, \widetilde{V}_{d_2}, \ldots, \widetilde{V}_{d_m}\}$.
- W — A keyword dictionary having n keywords, $W = \{w_1, w_2, \ldots, w_n\}$.
- PL — A list of keyword partitions, $PL = \{P_1, P_2, \ldots, P_\tau\}$.
- VF_{d_i} — The τ-dimensional DFB-vector of d_i.
- VF_D — The set of DFB-vectors of the documents, $VF_D = \{VF_{d_1}, VF_{d_2}, \ldots, VF_{d_m}\}$.
- Q — A query request with multi-keywords.
- V_Q — The n-dimensional query vector of Q.
- $\widetilde{V}_Q$ — The trapdoor of Q which is the encrypted n-dimensional query vector.
- VF_Q — The τ-dimensional QFB-vector of Q.
- CID — A set of candidate document IDs for the query Q.

B. PRELIMINARIES

1) VECTOR SPACE MODEL

The vector space model [19] adopting TF-IDF model [27] is widely adopted in secure multi-keyword search [5], [6], [11]–[13]. We also use such models in this paper. TF and IDF are the term frequency (TF) and inverse document frequency (IDF). The former is the number of times a given keyword or term exists in documents while the later is calculated by dividing the total number of all documents by the number of documents having the given keyword or term. Each document d_i is described by a n-dimensional vector where n is the scale of the keyword dictionary. $V_{d_i}[j]$ stores the normalized TF value of the keyword w_j as shown in Eq.(1). For a query Q having multiple searched keywords, the n-dimensional vector V_Q stores the normalized IDF values of the searched keywords in Q. The calculation of $V_Q[j]$ is shown in Eq.(2).

$$V_{d_i}[j] = TF_{d_i,w_j} / \sqrt{\sum_{w_j \in d_i \wedge d_i \in D} (TF_{d_i,w_j})^2} \tag{1}$$

$$V_Q[j] = IDF_{w_j} / \sqrt{\sum_{w_j \in Q} (IDF_{w_j})^2} \tag{2}$$

2) RELEVANCE SCORE MEASUREMENT

We adopt the same calculations in [13] to measure the relevance scores between documents and the search requests in this paper. We assume that d_i is a document and Q is a query request having multiple searched keywords, the relevance score between d_i and Q is calculated by the inner product between the corresponding document vector V_{d_i} and the query vector V_Q, i.e.

$$score(V_{d_i}, V_Q) = V_{d_i} \cdot V_Q = \sum_{j=1}^{n} V_{d_i}[j] \times V_Q[j] \tag{3}$$

3) SECURE INNER PRODUCT OPERATION

The secure inner product operation [20] is adopted in this paper. The operation is capable of computing the inner product of two encrypted vectors even their plaintext values are unknown. We assume that p and q are two n-dimensional vectors and M is a random $n \times n$-dimensional invertible matrix. Here, M is used as the secure key. We denote $\widetilde{p}$ and $\widetilde{q}$ as the the encrypted form of plaintext vectors p and q respectively. And $\widetilde{p}$ and $\widetilde{q}$ is calculated by $\widetilde{p} = pM^{-1}$ and $\widetilde{q} = qM^T$. Then we have

$$\begin{aligned} \widetilde{p} \cdot \widetilde{q} &= (pM^{-1}) \cdot (qM^T) \\ &= pM^{-1}(qM^T)^T \\ &= pM^{-1}Mq \\ &= p \cdot q \end{aligned} \tag{4}$$

Therefore, $\widetilde{p} \cdot \widetilde{q} = p \cdot q$ holds which indicates that the inner product of two encrypted vectors equals the inner product of the corresponding plaintext vectors.

4) NORMALIZED GOOGLE-DISTANCE

Given two keywords w_i and w_j, the Normalized Google-Distance [3] between them is denoted as $Dist(w_i, w_j)$ where

$$Dist(w_i, w_j) = \frac{max\{log_2 F_i, log_2 F_{i,j}\} - log_2 F_{i,j}}{log_2 m - min\{log_2 F_i, log_2 F_j\}} \tag{5}$$

In Eq.(5), F_i and F_j are the total frequencies of w_i and w_j appearing in the documents of D respectively, $F_{i,j}$ is the number of documents where w_i and w_j both appear in the same documents, m is the number of documents of D and $min\{X\}$ is to get the minimum from the set X. According to [3], the distance can be used to represent the relevance between keywords. The relevance between w_i and w_j increases as $Dist(w_i, w_j)$ decreases. The Normalized Google-Distance in the paper is only used for calculating the relevance between keywords.

IV. MODELS AND PROBLEM DESCRIPTION

A. SYSTEM MODEL

The system model adopted in this paper is the same as [26] which has four entities: the data owner (DO), the data user (DU), the private cloud (Pri-Cloud) and the public cloud (Pub-Cloud). The cooperation of them is shown in FIGURE 1.

1) DO owns the sensitive data. To protect the privacy of its data, DO encrypts documents and the corresponding vectors and then outsources the encrypted data in Pub-Cloud. DO constructs DFB-vectors as the index to speed the search

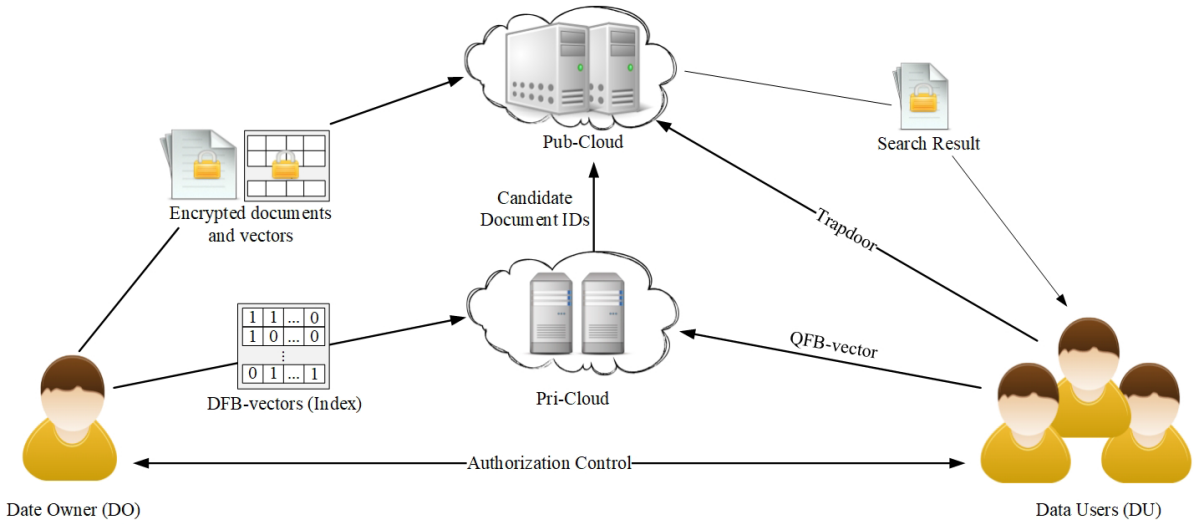

FIGURE 1. System model.

efficiency which are stored in Pri-Cloud. Besides, DO has the privilege to grant the authorization of accessing the outsourced data to DU.

2) DU is the user authorized by DO, who is authorized to search the data outsourced in Pub-Cloud. Once DU starts a ranked multi-keyword search, the queried keywords are transformed into a corresponding trapdoor and a QFB-vector which are submitted to Pub-Cloud and Pri-Cloud respectively for processing ranked searches. After DU receives the search result from Pub-Cloud, it decrypts the encrypted data to get the plaintext result.

3) Pri-Cloud is in charge of storing the index which are DFB-vectors of documents. Once receiving the QFB-vector of a query from DU, it performs the bitwise AND operations between the QFB-vector and DFB-vectors and filters out the candidate document IDs for the query and then transmits the IDs to Pub-Cloud for the further search processing.

4) Pub-Cloud is in charge of storing the outsourced data from DO. Once receiving the trapdoor and the candidate document IDs from DU and Pri-Cloud respectively, it performs the ranked search on the encrypted documents whose IDs are in the received candidate document IDs and then returns the result encrypted documents to DU.

B. SEARCH MODEL

Given a set of t queried keywords $Q = \{w_1, w_2, \ldots, w_t\}$, a multi-keyword ranked search is to retrieve the k ranked documents that are most relevant to Q. Formally, we define a multi-keyword search as $Query = (D, Q, k)$ where k is the number of requested documents and $k \ll |D|$ generally. For simplicity, we use the notation Q to represent the search. The result of the query Q, denoted as R, satisfies the following two conditions.

1) $|R| = k \wedge \forall d_i, d_j (d_i \in R \wedge d_j \in (D - R)) \rightarrow score(V_{d_i}, V_Q) \geq score(V_{d_j}, V_Q)$.

2) The documents in R are ranked according to the relevance scores between them and Q.

C. PROBLEM DESCRIPTION

In our scheme, we consider the same threat model as [26], which assumes that DO, DU and Pri-Cloud are trusted, but Pub-Cloud is considered as "honest-but-curious". It means that Pub-Cloud always processes the pre-deployed algorithms honestly and returns results correctly, but it is curious of peeping the plaintext of the outsourced data, which could cause privacy leakage through data analysis and deduction. We assume that Pub-Cloud has the encrypted data outsourced by DO, but it does not have the secure keys. In accordance with the background of Pub-Cloud, two threat models are adopted as follows, which are also adopted in many related works [5], [6], [11]–[13], [15], [17], [25].

- **Known Encryption Model.** In this model, Pub-Cloud only knows the encrypted documents $\widetilde{D}$ and the trapdoor $\widetilde{V}_Q$ but it does not have any plaintext information about them. It means that Pub-Cloud has to perform ciphertext-only attack (COA) [28] to observe the plaintext data.
- **Known Background Model.** In this model, Pub-Cloud is assumed to have more knowledge than the known ciphertext threat model, such as the keyword frequency statistics of document collection. The statistical information reveals the quantity of documents of specific keywords in D, which could be used by Pub-Cloud to apply TF statistical attacks and hence infer or even recognize certain keywords through analyzing the histogram or value range of the corresponding frequency distributions [11], [12], [29].

In this paper, we focus on the multi-keyword ranked search scheme over encrypted data in hybrid clouds. The designed goals are as follows.

1) MULTI-KEYWORDS RANKED SEARCH

The proposed scheme is designed that Pub-Cloud can determine the ranked k encrypted documents which have the k

highest relevance scores to the searched keywords through the cooperation with Pri-Cloud.

2) SEARCH EFFICIENCY

The proposed scheme is able to perform efficient multi-keyword ranked searches by using a special index constructed on the basis of the given keyword partition vector model and the complete binary tree structure. The index can filter out candidate documents and prune a large number of irrelevant documents.

3) PRIVACY-PRESERVING

The proposed scheme is able to preserve privacy from the curious Pub-Cloud. Particularly, the plaintext of documents, index and queried keywords should be kept in private and the trapdoor unlinkability [5], [6], [11]–[13] should be protected.

V. KEYWORD PARTITION VECTOR MODEL

To describe the keyword partition vector model (KPVM), the clustering based keyword partition algorithm is first introduced in this section. Then the keyword partition based bit vectors are defined formally, which are the index of the proposed scheme.

A. CLUSTERING BASED KEYWORD PARTITION

We design the algorithm *GenPartitions* to partition the keyword dictionary W which is on the basis of the bisecting k-means clustering [30] and shown in Algorithm 1. The Normalized Google-Distance [3] is adopted to measure the distance between keywords. A partition list, denoted as $PL = \{P_1, P_2, \ldots, P_\tau\}$, is the output of this algorithm where τ is a threshold to control the number of partitions.

Algorithm 1 *GenPartitions*(W, τ)

Input: The keyword dictionary W.
Output: The keyword partition list PL.
1: Initialize $PL = \emptyset$;
2: Add W to PL where W is treated as a keyword partition;
3: **while** $|\mathrm{PL}| < \tau$ **do**
4: $P_{max} = max(PL)$;
5: Apply the bisecting k-means clustering algorithm to the partition P_{max} by using the Normalized Google-Distance of keywords, and then append the generated two keyword clusters as two partitions in PL;
6: **end while**
7: **return** PL

In Algorithm 1, $max(PL)$ is the function to get the biggest partition of PL which has the most documents. In each round of bisecting k-means clustering, the biggest partition of PL is divided into two smaller partitions. Therefore, Algorithm 1 is a balanced keyword partition algorithm which tends to balance the number of keywords of generated partitions even with different default parameter setting of clustering algorithm. Meanwhile, keywords with high relevance are clustered into partitions because of the adoption of the bisecting k-means clustering and the Normalized Google-Distance.

In addition, there are two reasons for the selection of the bisecting k-means algorithm. First, the bisecting k-means algorithm has the advantage of processing documents, comparing with other clustering algorithm like DBSCAN [31]. The second reason for choosing the bisecting k-means algorithm is that it can generate the balanced clusters.

Observation 1: According to Algorithm 1, we have the following two properties about the generated keyword partitions.

(1) $\bigcup_{P_i \in PL} P_i = W$
(2) $\forall P_i, P_j \in PL \rightarrow P_i \cap P_j = \emptyset$

Observation 1 can be deduced from the procedures of Algorithm 1. This observation indicates that the generated partitions are the divisions of the keyword dictionary and there are no intersections between any two partitions.

Definition 1 (Involved Partitions): Given a document $d_i \in D$, the involved partitions of d_i are the partitions that have at least a keyword of d_i. We denote the set of involved partitions of d_i as $IPS(d_i)$, then we have

$$IPS(d_i) = \{p_j | p_j \cap d_i \neq \emptyset \wedge P_j \in PL\} \tag{6}$$

Definition 2 (Covered Documents): Given a keyword partition $P_i \in PL$, the covered documents of P_i are the documents that have at least a keyword of P_i. We denote the set of covered documents of P_i as $CDS(P_i)$, then we have

$$CDS(P_i) = \{d_j | d_j \cap P_i \neq \emptyset \wedge d_j \in D\} \tag{7}$$

According to Definition 1 and 2, we can deduce Observation 2 as follows.

Observation 2: Given a document d_i and a keyword partition P_j, if d_i is in the covered documents of P_j, then P_j is in the involved partitions of d_i, and vice versa.

$$d_i \in CDS(P_j) \leftrightarrow P_j \in IPS(d_i) \tag{8}$$

Definition 3 (Target Partitions): Given a query Q with multi-keyword, the target partitions of Q is the keyword partitions that have at least one queried keywords of Q. We denote the set of target partitions of Q as $TPS(Q)$, then we have

$$TPS(Q) = \{P_i | Q \cap P_i \neq \emptyset \wedge P_i \in PL\} \tag{9}$$

Definition 4 (Candidate Documents): Given a query Q, the candidate documents of Q are the covered documents of the target partitions of Q. We denote the candidate documents of Q as $CDocs(Q)$, then we have

$$CDocs(Q) = \bigcup_{P_i \in TPS(Q)} CDS(P_i) \tag{10}$$

We give the example to describe the above algorithm and definitions.

Example 1: FIGURE 2 illustrates an example of keyword partitions generated by *GenPartitions*. As shown in the figure,

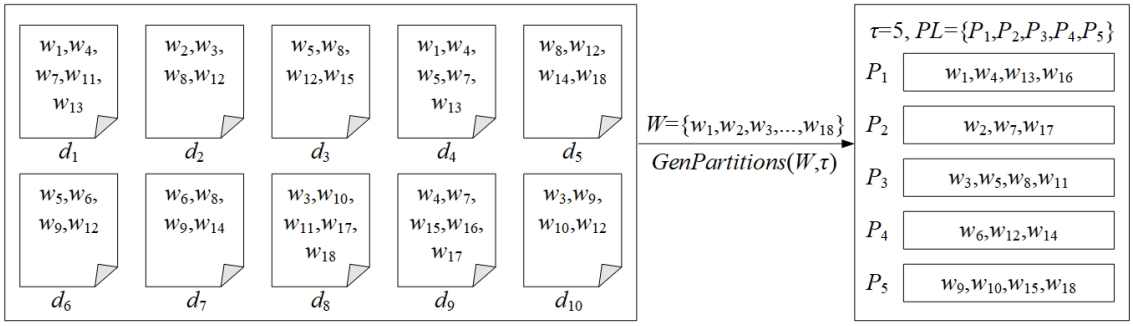

FIGURE 2. Example of *GenPartitions*.

we assume that document collection D has 10 documents and the threshold of keyword partitions e.g. τ is set to be 5. The list of keyword partitions is $PL = \{P_1, P_2, P_3, P_4, P_5\}$ and the detailed descriptions are shown in the figure. According to Definition 1, the involved partitions of D are described as $IPS(d_1) = \{P_1, P_2, P_3\}$, $IPS(d_2) = \{P_2, P_3, P_4\}$, $IPS(d_3) = \{P_3, P_4, P_5\}$, etc. According to Definition 2, the covered documents of P_1, P_2, P_3, P_4 and P_5 are expressed as $CDS(P_1) = \{d_1, d_4, d_9\}$, $CDS(P_2) = \{d_1, d_2, d_4, d_9\}$, etc. Suppose the query $Q = \{w_1, w_4, w_7, w_{16}\}$, according to Definition 3 and 4, then we have the target partitions $TPS(Q) = \{P_1, P_2\}$ and the corresponding candidate documents $CDocs(Q) = CDS(P_1) \cup CDS(P_2) = \{d_1, d_2, d_4, d_9\}$. Obviously, the query result must be in $CDocs(Q)$ and the other documents could be out of consideration directly.

B. KEYWORD PARTITION BASED BIT VECTORS

Definition 5 (Document Filtering Bit Vector (DFB-Vector)): Given a document $d_i \in D$, the DFB-vector of d_i is a τ-dimensional bit vector which is denoted as VF_{d_i}. If there is a keyword in d_i belongs to a partition $P_j \in PL$, then $VF_{d_i}[j] = 1$ otherwise $VF_{d_i}[j] = 0$, i.e.

$$VF_{d_i}[j] = \begin{cases} 1, & \exists w_P \in d_i (w_P \in P_j) \\ 0, & Else \end{cases} \quad j \in \{1, 2, \ldots, \tau\} \quad (11)$$

Definition 6 (Query Filtering Bit Vector (QFB-Vector)): Given a query Q with multiple keywords, the QFB-vector of Q is τ-dimensional bit vector which is denoted as VF_Q. If there is a keyword in Q belongs to a partition $P_i \in PL$, then $VF_Q[i] = 1$ otherwise $VF_Q[i] = 0$, i.e.

$$VF_Q[i] = \begin{cases} 1, & \exists w_P \in Q (w_P \in P_i) \\ 0, & Else \end{cases} \quad i \in \{1, 2, \ldots, \tau\} \quad (12)$$

According to Definition 5 and 6, we have that the DFB-vector indicates the involved partitions of the corresponding document while the QFB-Vector indicates the target partitions of a query. For example, if $VF_{d_i}[j] = 1$, then P_j is a involved partition of d_i, which means that d_i is a covered document of P_j. And if $VF_Q[j] = 1$, then P_j is the target partition of the query Q and d_i is a candidate document of Q. Therefore, we can deduce Observation 3 as follows.

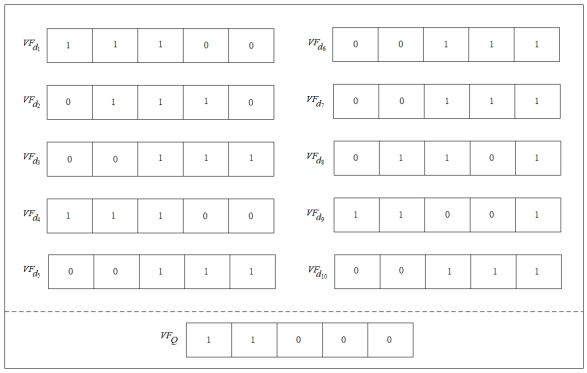

FIGURE 3. VF_D and VF_Q.

Observation 3. Given a query Q, we have

$$CDocs(Q) = \{d_i | VF_{d_i} \& VF_Q \neq \{0\}^\tau \wedge d_i \in D\} \quad (13)$$

where "&" is the bitwise AND operator and $\{0\}^\tau$ represents a τ-dimensional zero bit vector.

Observation 3 indicates that, given a document d_i, if the bitwise AND operation result between the DFB-vector of d_i and the QFB-vector of Q is not a zero bit vector, then d_i is a candidate document of Q. Therefore, the DFB-vectors and QFB-vector are the index for filtering out the candidate documents and speeding up the searches.

We also take Example 1 as an example where ten documents and the query $Q = \{w_1, w_4, w_7, w_{16}\}$ are assumed. According to Definition 5 and 6, the DFB-vectors of the documents and the QFB-vector of Q are shown in FIGURE 3, which are all 5-dimensional vectors. According to Observation 3, we apply the bitwise AND operation between the DFB-vectors and the QFB-vector, then we have $CDocs(Q) = CDS(P_1) \cup CDS(P_2) = \{d_1, d_2, d_4, d_9\}$ which coincides the result of Definition 4.

VI. MRSE-HC SCHEME

Based on the keyword partition vector model, we give the details procedures of our scheme. We first give the framework of MRSE-HC shown in FIGURE 4. It consists of two stages which are setup stage and search stage. Before providing ranked search services, DO performs the algorithms, *GenKey*, *GenPatitions*, *GenVector*, *EncData* and *StoreData*, to set up the system. Once a query is applied, the search stage is

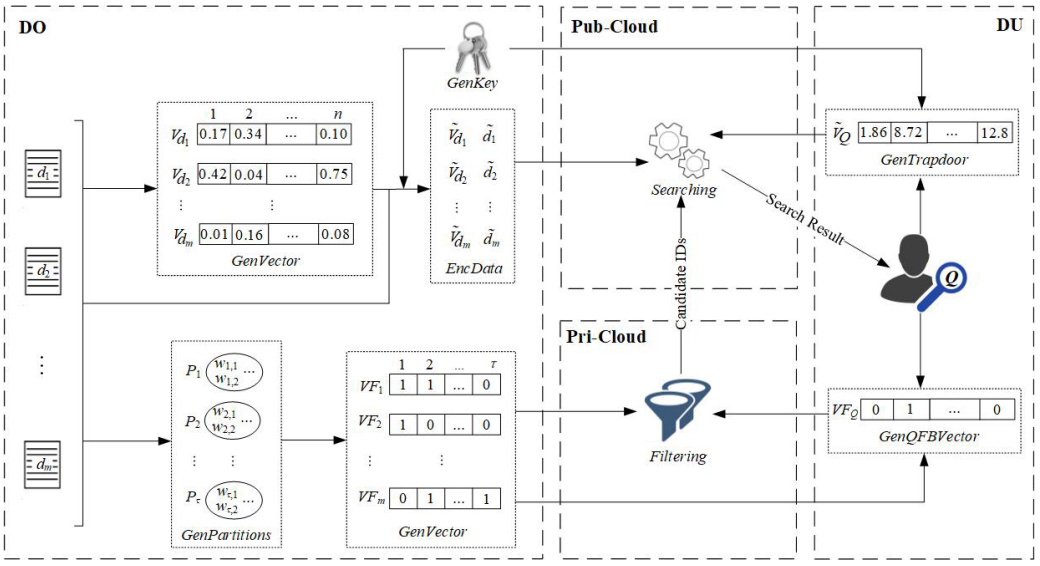

FIGURE 4. The framework of MRSE-HC.

performed by the algorithms, *GenTrapdoor*, *GenQFBVector*, *Filtering* and *Searching*. Detailed statements about the above algorithms are then given in the following two sub-sections.

A. ALGORITHMS IN SETUP STAGE

1) $SK \leftarrow GenKey(1^{l(n)})$

DO generates the secure key $SK = \{S, M_1, M_2, g\}$ where S is a random generated n-dimensional bit vector, M_1 and M_2 are random $n \times n$-dimensional invertible matrices, and g is the key for document encryption. SK is shared by DO and DU.

2) $PL \leftarrow GenPartitions(W, \tau)$

This algorithm is given in Algorithm 1 in detail. It is on the basis of the bisecting k-means clustering and the Normalized Google-Distance measurement. After performing the algorithm, keywords with high relevance in the keyword dictionary are clustered into partitions and the partition list $PL = \{P_1, P_2, \ldots, P_\tau\}$ is generated where τ is a threshold to control the number of output partitions.

3) $\{V_D, VF_D\} \leftarrow GenVectors(D, PL)$

For each $d_i \in D$, DO generates the corresponding document vector V_{d_i} according to Eq.(1), and then generates the corresponding DFB-vector VF_{d_i} according to Definition 5. The sets of generated document vectors and DFB-vectors are $V_D = \{V_{d_1}, V_{d_2}, \ldots, V_{d_m}\}$ and $VF_D = \{VF_{d_1}, VF_{d_2}, \ldots, VF_{d_m}\}$ respectively. Here, the generated DFB-vectors will be utilized as the index in our scheme to filter out the candidate documents and speed the ranked searches.

4) $\{\widetilde{D}, \widetilde{V}_D\} \leftarrow EncData(D, V_D, SK)$

For each $d_i \in D$ and the corresponding document vector $V_{d_i} \in V_D$, DO first encrypts d_i into $\widetilde{d}_i$ by a symmetric encryption (such as DES, AES, *et al.*) with the secret key g in SK. Second, DO generates two random n-dimensional vectors $\{V_{d_i}^1, V_{d_i}^2\}$ according to the random bit vector S in SK. Specifically, if $S[j] = 0$, then $V_{d_i}^1[j] = V_{d_i}^2[j] = V_{d_i}[j]$; otherwise $V_{d_i}^1[j] = GenRand()$ and $V_{d_i}^2[j] = V_{d_i}[j] - V_{d_i}^1[j]$ where $GenRand()$ is a random value generator. Then, the encrypted document vector $\widetilde{V}_{d_i}$ is calculated, $\widetilde{V}_{d_i} = \{V_{d_i}^1 M_1^T, V_{d_i}^2 M_2^T\}$. Through the above operations, the encrypted documents $\widetilde{D} = \{\widetilde{d}_1, \widetilde{d}_2, \ldots, \widetilde{d}_m\}$ and the corresponding encrypted document vectors $\widetilde{V}_D = \{\widetilde{V}_{d_1}, \widetilde{V}_{d_2}, \ldots, \widetilde{V}_{d_m}\}$ are generated.

5) $StoreData(VF_D, \widetilde{D}, \widetilde{V}_D)$

After the above steps, DO outsources the encrypted documents $\widetilde{D}$ and the corresponding encrypted document vectors $\widetilde{V}_D$ to Pub-Cloud. And then DO uploads the DFB-vectors VF_D to Pri-Cloud. It is noticeable that the corresponding document IDs are both stored in Pub-Cloud and Pri-Cloud. Obviously, the shared data between Pub-Cloud and Pri-Cloud are only the document IDs.

After the five steps, the setup stage is finished and the system is prepared for the multi-keyword search over encrypted documents.

B. ALGORITHMS IN SEARCH STAGE

1) $\widetilde{V}_Q \leftarrow GenTrapdoor(Q, SK)$

Once a query Q with multi-keywords is applied, DU generates the query vector V_Q according to Eq.(2) and two random n-dimensional vectors $\{V_Q^1, V_Q^2\}$ according to the random bit vector S in SK. Specifically, if $S[i] = 1$, then $V_Q^1[i] = V_Q^2[i] = V_Q[i]$; otherwise $V_Q^1[i] = GenRand()$ and $V_Q^2[i] = V_Q[i] - V_Q^1[i]$. Then, the encrypted query vector $\widetilde{V}_Q$ is calculated, $\widetilde{V}_Q = \{V_Q^1 M_1^{-1}, V_Q^2 M_2^{-1}\}$, which is the trapdoor of Q and submitted to Pub-Cloud.

2) $VF_Q \leftarrow GenQFBVector(Q, PL)$

According to Definition 6, DU generates the QFB-vector of Q, VF_Q, which indicates the target partitions of Q. And then DU transmits VF_Q to Pri-Cloud.

3) $CID \leftarrow Filtering(VF_Q, VF_D)$

Pri-Cloud utilizes the DFB-vectors as the index to filter out the candidate documents for the query. It performs the bitwise AND operation between each DFB-vector and the received QFB-vector and then find out the corresponding IDs of the candidate documents for the query Q according to Observation 3. Specifically, for each $VF_{d_i} \in VF_D$, if $VF_{d_i} \& VF_Q$ is not a zero-vector, then $Identity(d_i)$ is added in the ID set CID. Here, $Identity(d_i)$ is the ID of d_i. After processing all the DFB-vectors of VF_D, Pri-Cloud only transmits CID to Pub-Cloud. CID is the only shared data between Pri-Cloud and Pub-Cloud. The complexity of this algorithm is $O(m*\tau)$ since there are $m * \tau$ bitwise AND operations.

4) $\Re \leftarrow Searching(\widetilde{D}, \widetilde{V}_D, \widetilde{V}_Q, CID, k)$

Pub-Cloud computes the inner products between the trapdoor $\widetilde{V}_Q$ and the encrypted document vectors $\{\widetilde{V}_{d_i} | \widetilde{V}_{d_i} \in \widetilde{V}_D \wedge Identity(d_i) \in CID\}$. According to Lemma 1, the inner products between the trapdoor and the encrypted document vectors equal the inner products between the corresponding plaintext query vector and document vectors respectively, and the inner products represent the relevance scores between the queried keywords and the documents according to the relevance score measurement in the Preliminaries Section. The ranked k encrypted documents with the highest k inner products are the result encrypted documents R which is returned to DU. The complexity of this algorithm is $O(|CID| * n + k * |CID|) \approx O(|CID| * n)$ since $k \ll n$ holds generally. $|X|$ represents the item number of the set X.

At last, when DU receives the result encrypted documents $\Re$ from Pub-Cloud, it uses the shared secure key to decrypts the encrypted documents and get the plaintext result documents.

Lemma 1: $\widetilde{V}_Q \cdot \widetilde{V}_{d_i} = V_Q \cdot V_{d_i}$

Proof:

$$\begin{aligned}\widetilde{V}_Q \cdot \widetilde{V}_{d_i} &= \{V_Q^1 M_1^{-1}, V_Q^2 M_2^{-1}\} \cdot \{V_{d_i}^1 M_1^T, V_{d_i}^2 M_2^T\}^T \\ &= (V_Q^1 M_1^{-1}) \cdot (V_{d_i}^1 M_1^T)^T + (V_Q^2 M_2^{-1}) \cdot (V_{d_i}^2 M_2^T)^T \\ &= V_Q^1 M_1^{-1} M_1 (V_{d_i}^1)^T + V_Q^2 M_2^{-1} M_2 (V_{d_i}^2)^T \\ &= V_Q^1 (V_{d_i}^1)^T + V_Q^2 (V_{d_i}^2)^T \\ &= V_Q \cdot V_{d_i}\end{aligned}$$

■

In the above algorithms, DU transforms the query request into two vectors, one is the trapdoor (the encrypted query vector) generated in *GenTrapdoor* and the other is the QFB-vector generated in *GenQFBVector*. The latter is submitted to Pri-Cloud and Pri-Cloud uses it and the DFB-vectors of documents to filter out the candidate document IDs in *Filtering*. According to the received candidate document IDs, Pub-Cloud determines the result encrypted documents and return them to DU in *Searching*. Since the searching space is shrunk in the candidate documents, the ranked search efficiency is improved. We will give the performance evaluation on search efficiency in Section *VIII*.

VII. THE ENHANCED SCHEME

In this section, we propose the enhanced scheme EMRSE-HC which is designed to improve the efficiency of MRSE-HC. EMRSE-HC utilizes the complete binary pruning tree (CBP-Tree) instead of the sequential DFB-vectors and the corresponding CBP-Tree based filtering algorithm is adopted. Compared with MRSE-HC, EMRSE-HC adds the CBP-Tree construction algorithm *BuildCBPT* and updates the *StoreData* and *Flitering* algorithms. The three algorithms are briefly introduced as follows.

- $\mathcal{I} \leftarrow BulidCBPT(D)$. In EMRSE-HC scheme, the CBP-Tree, denoted as $\mathcal{I}$, is constructed as index. Each node in $\mathcal{I}$ stores the DFB-vector and the pruning vector, the latter of which is used for improving the search efficiency. Details of this algorithm is illustrated in Section *VII.A*.
- $StoreData(\mathcal{I}, \widetilde{D}, \widetilde{V}_D)$. In EMRSE-HC scheme, $\mathcal{I}$ is stored in Pri-Cloud. And the storage of $\widetilde{D}$ and $\widetilde{V}_D$ are the same as MRSE-HC.
- $CID \leftarrow Filtering(\mathcal{I}, i, VF_Q)$. In EMRSE-HC scheme, the *Filtering* algorithm utilizes the CBP-Tree to filter out the unqualified documents efficiently and generates the candidate documents. Details of the updated *Filtering* algorithm is shown in Section *VII.B*.

A. CBP-TREE CONSTRUCTION ALGORITHM

The CBP-Tree is the index of EMRSE-HC, which is a complete binary tree. Each node corresponds to a document. The data structure of the node is defined as:

$$< docID, dfv, pv >, \tag{14}$$

where $docID$ and dfv are the identity and DFB-vector of a document respectively, pv is the pruning vector of the node which is used for filtering out unqualified nodes.

According to [32], we know that an array is an appropriate structure to store a complete binary tree. Thus, we utilize the array $\mathcal{I}[1, 2, \ldots, m]$ to represent the CBP-Tree. The conclusion of [32] show that for the node $\mathcal{I}[i]$, if $i = 1$, $\mathcal{I}[i]$ is the root node; if $i > 1$, the parent node of $\mathcal{I}[i]$ is denoted as $\mathcal{I}[\lfloor i/2 \rfloor]$; if the left and right children of $\mathcal{I}[i]$ are $\mathcal{I}[2i]$ and $\mathcal{I}[2i + 1]$ respectively if they exist. The CBP-Tree construction algorithm is shown in Algorithm 2.

In Algorithm 2, if the node $\mathcal{I}[i]$ is a leaf node, its pruning vector is set to equal the DFB-vector of the node. If $\mathcal{I}[i]$ is an internal node, the pruning vector of $\mathcal{I}[i]$ is generated by the bitwise OR operation, depending on whether it has right child node. Since the CBP-Tree is a complete binary tree with m nodes, the height is $\lceil log_2(m + 1) \rceil$ which is the shortest of the binary trees with m nodes.

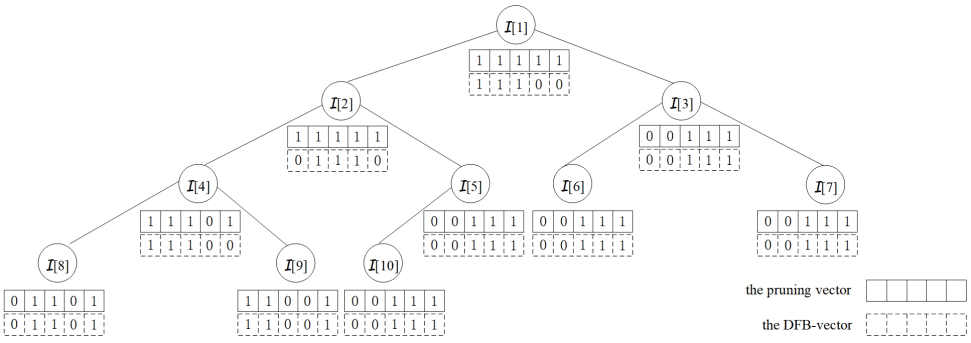

FIGURE 5. CBP-Tree.

Algorithm 2 *BulidCBPT*$(D, \mathcal{I})$

Input: The document collection D.
Output: CBP-Tree $\mathcal{I}[1, 2, \ldots, m]$ where $\mathcal{I}[i]$ is a node of the tree.

1: **for** $i = 1, 2, \ldots, m$ **do**
2: Settle the node $\mathcal{I}[i]$ where $\mathcal{I}[i].docID = id(d_i)$, $\mathcal{I}[i].pv = \mathcal{I}[i].dfv = VF_{d_i}$;
3: **end for**
4: **for** $i = \lceil (m-1)/2 \rceil, \ldots, 1$ **do**
5: **if** $2i + 1 \leq n$ **then**
6: $\mathcal{I}[i].pv = \mathcal{I}[i].pv \,|\, \mathcal{I}[2i].pv \,|\, \mathcal{I}[2i+1].pv$;
7: **else**
8: $\mathcal{I}[i].pv = \mathcal{I}[i].pv \,|\, \mathcal{I}[2i].pv$;
9: **end if**
10: **end for**
11: **return** $\mathcal{I}$

We take the same example as Example 1 to illustrate Algorithm 2. The DFB-vectors of documents in Example 1 is shown in FIGURE 4. Taking those DFB-vectors as the input of Algorithm 2 and the constructed CBP-Tree is shown in FIGURE 5.

B. CBP-TREE BASED FILTERING ALGORITHM

DO stores the CBP-Tree to Pri-Cloud. Pri-Cloud utilizes the pruning vectors of nodes in the CBP-Tree to prune unqualified subtrees. The CBP-Tree based Filtering algorithm is shown in Algorithm 3.

Algorithm 3 is a recursion algorithm that starts with the root node. Algorithm 3 can find all candidate nodes which are coinciding the result of bitwise AND operation between its DFB-vector and QFB-vector is not equal to 0. DFB-vector in each node is utilized to early prune unqualified nodes. The pruning vector is used to improve the filter efficiency. Because the height of the CBP-Tree is $log_2 m$ and the unqualified nodes are early filtered out, the complexity of this algorithm is $O(\tau * log_2 m)$.

The enhanced *Filtering* algorithm improves the filtering efficiency and promotes the search efficiency. Additionally, we give the detailed performance evaluation in Section *VIII*.

Algorithm 3 *Filtering*$(\mathcal{I}, i, VF_Q, CID)$

Input: $\mathcal{I}[i]$ is the current processed node, VF_Q is the QFB-vector.
Output: *CID* is the candidate documents ID collection.

1: **if** $i \leq n$ **then**
2: **if** $2i > n$ **then**
3: **if** $\mathcal{I}[i].dfv \,\&\, VF_Q \neq 0$ **then**
4: Add $\mathcal{I}[i].docID$ into *CID*;
5: **end if**
6: **else**
7: **if** $\mathcal{I}[i].pv \,\&\, VF_Q \neq 0$ **then**
8: **if** $\mathcal{I}[i].dfv \,\&\, VF_Q \neq 0$ **then**
9: Add $\mathcal{I}[i].docID$ into *CID*;
10: **end if**
11: *Filtering*$(\mathcal{I}, 2i, VF_Q, CID)$;
12: *Filtering*$(\mathcal{I}, 2i+1, VF_Q, CID)$;
13: **end if**
14: **end if**
15: **end if**

C. SECURITY ENHANCEMENT

In the search stage, the Pub-Cloud generates different trapdoors when the same queries are applied but the candidate documents and the calculated relevance scores are the same. Pub-Cloud could use those convert channels to link the same search requests and deduce the hot keywords with high frequency in documents. To overcome this problem, it is a practical and effective countermeasure to extend dimension by adding some phantom terms into vectors to break such convert channels [5], [6], [11], [13]. We also introduce this method to enhance the security of our scheme and to protect the document confidentiality, index and trapdoor privacy and trapdoor unlinkability.

We adopt the similar phantom item addition method of [5], [6], [11], [13] for providing trapdoor unlinkability. The brief idea is introduced as follows. First, DO randomly generates two $(n + L + 1)$-dimensional vectors by S' and two $(n + L + 1) \times (n + L + 1)$-dimensional invertible matrices $\{M_1', M_2'\}$. Then, the document vectors and query vectors are extended to $(n + L + 1)$-dimensional vector where L is the number

dummy keywords inserted. The $(n + j + 1)$-th entry in the extended document vector V'_{d_i} is set to a random number ε_j where $j \in \{1, 2, \ldots, L\}$ and the $(n + L + 1)$-th is set to 1. Finally, by randomly selecting G out of L dummy keywords and the corresponding values in the extended query vector V'_Q are set to 1 and then V'_Q multiple by a random parameter r, the $(n + L + 1)$-th is set to a random entry λ. The above extended vectors are used for computing the ranked encrypted documents. Due to the adoption of the secure inner product operation, the phantom items are hidden in the encrypted extended vectors and it is hard to distinguish between the phantom items and the real items. What is noteworthy is that some added phantom terms will slightly decrease the accuracy of the query result but the trapdoor unlinkability is preserved.

Adding the phantom item in the document vectors and the query vectors affects the accuracy of the relevance score and the query results. But the trapdoor unlinkability is preserved. In addition, the standard deviation parameter σ could be adjusted to balance the accuracy and the trapdoor unlinkability. Analysis of the effects of adding phantom items can be referred in [6], [11].

D. SECURITY ANALYSIS

In this section, we analyze the EMRSE-HC scheme according to the three privacy demands.

1) DOCUMENT CONFIDENTIALITY

In the scheme, the documents and the corresponding vectors are encrypted and then outsourced in Pub-Cloud. The encryption procedures are given in the *EncData* algorithm. Documents are encrypted by a symmetric encryption algorithm (such as AES) with the secret key g in SK while the corresponding vectors are encrypted by the secure inner product operation with one random bit vector and two random invertible matrix in SK. Since SK generated by DO is only shared with the authorized DU, Pub-Cloud has no idea of those secret keys in SK. Thus, it is computation infeasible for Pub-Cloud to obtain the plaintext information from the encrypted documents and vectors and document confidentiality is guaranteed.

2) INDEX AND TRAPDOOR PRIVACY

In the scheme, the index is the CBP-Tree stored in Pri-Cloud which is isolated from Pub-Cloud. Trapdoors are generated by performing secure inner product operation on the query vectors with the secret key SK which is only shared by DO and DU. Additionally, several random items are added in the trapdoor generation process, which increases the randomness of values in vectors. Without the secret key and the phantom item addition method, it is hard for Pub-Cloud to get the plaintext query vectors. As a result, the scheme can protect the privacy of the index and the trapdoor.

TABLE 1. Default values of parameters.

Parameters	τ	k	t	m	n
Values	200	15	15	30000	6000

3) TRAPDOOR UNLINKABILITY

In this scheme, the probability that *GenTrapdoor* algorithm generates the same trapdoor by the same query is extremely low and can be considered negligible. We analyze the reason under the known background model. According to the phantom item addition procedures in Section *VII.C*, the original query vector is extended into $n + L + 1$-dimensional, has $n + L + 1$ bits after adding $L + 1$ dimensions in the o for adding phantom items. r is the noise parameter for enhancing trapdoors by performing multiplication between r and each dimension of trapdoor. The length of r is denoted as l_r. The former n bits of the extended trapdoor has 2^n possibilities.

The possibility of the same query generates the same trapdoor $p = \frac{1}{x \times 2^{l_r}}$, and the larger the denominator, the smaller the p. In other words, the bigger the parameters, the smaller the probability of generating the same trapdoor, the safer the scheme. For example, we assume r is a 1024 bit parameter, then we have $p < \frac{1}{2^{1024}}$ which is considered negligible. Therefore, trapdoor is unlinkability in this scheme.

VIII. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our proposed basic scheme MRSE-HC, efficiency enhanced scheme EMRSE-HC and compare them with the scheme presented in [26] which is denoted as FMRS. We implement MRSE-HC, EMRSE-HC and FMRS and perform the evaluations on the search time cost on the real data set of NSF Research Award Abstracts provided by UCI [33]. The real dataset includes about 129000 abstracts. We use IK Analyzer [34] to extract the keywords of documents and then process the extracted keywords.

The experimental hardware environment is INTEL Core(TM) I5-8250 CPU, 4G memory, and 588G hard disk; and software environment is Eclipse development platform. Since the queried keywords are usually relevant with each other in practice, we assume that the default number of target partitions of a query is one. Other default parameters are summarized in Table 1 where m, n, τ, t and k are the number of documents, keywords in the dictionary, clustered partitions, queried keywords and request documents respectively.

In the following experiments, we evaluate the time cost of searches where one of the above parameters changes and the other parameters adopt the default values. The results are shown in FIGURE 6-10.

FIGURE 6-10 all show that the proposed MRSE-HC and EMRSE-HC both outperform FMRS in the time cost of ranked searches. Among them, EMRSE-HC is the most efficiently. On average, EMRSE-HC saves about 20% and 80% of the time cost respectively. The reason is that: (1) The target partitions of MRSE-HC and EMRSE-HC are usually

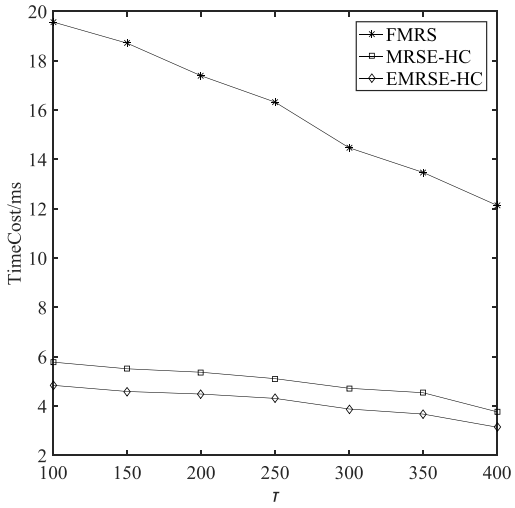

FIGURE 6. Search time cost vs τ.

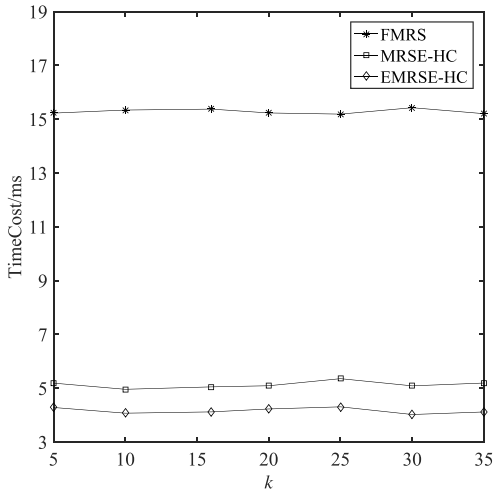

FIGURE 7. Search time cost vs k.

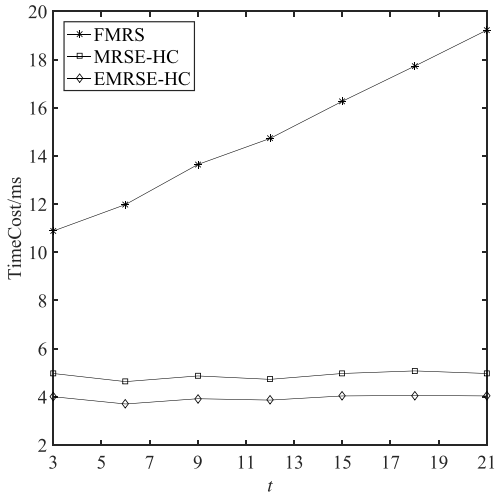

FIGURE 8. Search time cost vs t.

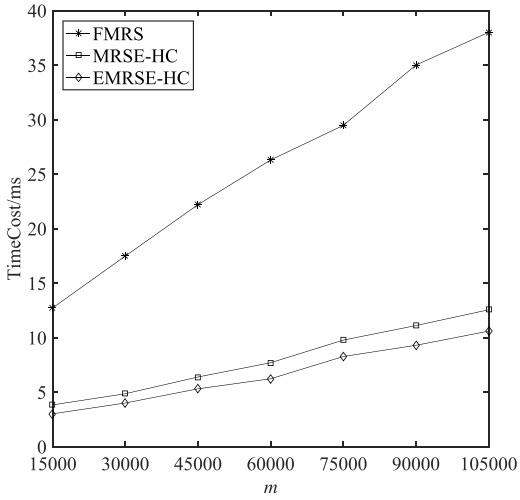

FIGURE 9. Search time cost vs m.

less than FMRS when a query is applied because the partitions generated in MRSE-HC or EMRSE-HC is clustered according to the relevance between keywords and the queried keywords are usually relevant to each other. The less are the target partitions, the less are the candidate documents determined, which improves the search efficiency. (2) In EMRSE-HC, the tree index, CBP-Tree is adopted and a large number of irrelevant documents are filtered out. Thus, the search efficiency is improved.

We analyze the impacts of the changes of τ, k, t, m and n on the search time cost of MRSE-HC, EMRSE-HC and FMRS as follows.

1) FIGURE 6 indicates that, as τ grows up, the time cost of MRSE-HC, EMRSE-HC and FRMS all decrease. The reason is that, when τ grows up, more partitions are generated, and the keywords in each partition and the covered documents of partitions both decrease. Since the target partitions are settled constantly, the total covered documents of the target partitions, which are the candidate documents, decrease simultaneously. Thus, the time cost of both schemes decreases.

2) FIGURE 7 shows that the increase of the number of request documents k have little impact on the time cost of MRSE-HC, EMRSE-HC and FMRS. The reason is that the search conditions (such as partitions, vectors and documents) remain unchanged as k growing up, thus the candidate documents in both schemes are still the same and the time cost of both schemes just randomly oscillate around a certain level.

3) FIGURE 8 indicates that as the number of queried keywords t grows up, the time cost of FMRS increases while the time cost of MRSE-HC and EMRSE-HC just have random oscillations. The reason is that keywords in FMRS are equally divided and randomly distributed in partitions while keywords in MRSE-HC and EMRSE-HC are clustered with high relevance in partitions. When t grows up, the number of target partitions proportionally increases in FMRS. But it has slight changes in MRSE-HC and EMSE-HC. Since the corresponding candidate documents have proportional relations with the target partitions, the time cost of FMRS increases while the time cost of MRSE-HC and EMRSE-HC just have slight changes.

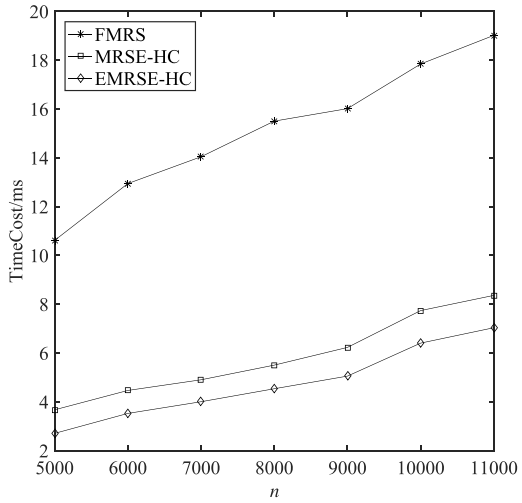

FIGURE 10. Search time cost vs n.

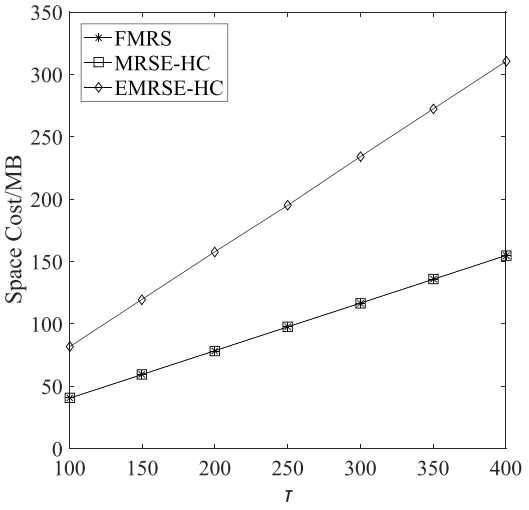

FIGURE 11. Space cost of index vs τ.

4) FIGURE 9 indicates that the time cost of MRSE-HC, EMRSE-HC and FMRS all increase as m grows up. The reason is that, when the number of documents m increases, the average number of covered documents of each partition grows up simultaneously. Since the target partitions of both schemes do not change, the total covered documents of the target partitions increase. Thus the candidate documents increase which consumes more time cost to perform the ranked searches.

5) FIGURE 10 shows that, as n grows up, the time cost of MRSE-HC, EMSE-HC and FMRS increase. The reason is that the average number of keywords in each partition grows up as the dictionary capacity n increases, which causes the covered documents of each partition increase. Since the target partitions of both schemes have no changes, the candidate documents (which is the covered documents of the target partitions) increase simultaneously. In addition, the dimensions of document vectors and query vectors are both enlarged when n grows up. It will consume more time in the relevance score calculations. Thus, all schemes consume more time to accomplish ranked searches.

6) FIGURE 11 shows that the space cost of the indexes in MRSE-HC, EMRSE-HC and FMRS all increase as τ grows up. The reason is that, when τ grows up, more partitions are generated and the dimensions of document filtering bit vectors increase simultaneous. The space cost of indexes in MRSE-HC and FMRS are the same. The reason is that the keywords in MRSE-HC and FMRS are the same thus the vectors in the indexes of them are also the same. In addition, the space cost of index in EMRSE-HC is about twice as it in MRSE-HC. The reason is that the index in EMRSE-HC is a complete binary tree, where the total nodes are nearly twice as the leaf nodes. Here, the leaf nodes store the same vectors of MRSE-HC and the internal nodes store the same structure of vectors as the leaf nodes.

IX. CONCLUSION

It is still a challenge to ensure the efficiency of the search under the premise of ensuring the accuracy of the search results. And most of the existing multi-keyword ranked search over encrypted data are for the public cloud. In this paper, we propose a privacy-preserving Multi-Keyword Ranked Search over Encrypted data in hybrid clouds,which is denoted as MRSE-HC. In our scheme, the keyword dictionary of documents are balancing clustered in partitions by the keyword partition algorithm on the basis of the bisecting k-means clustering. Based on the generated keywords partitions, construct the DFB-vector indicates the involved partitions of the corresponding document and the QFB-Vector indicates the target partitions of a query. And then Pri-Cloud uses the QFB-vector and DFB-vectors to find the candidate document IDs which can efficiently cut most irrelevant documents. Finally, Pub-Cloud uses the IDs and the query trapdoor to determine the result encrypted documents and returns them to users. Besides, we utilize secure inner product algorithm against two threat models. The experimental results show that the scheme proposed in this paper has better performance in terms of efficiency compared with the existing methods.

ACKNOWLEDGMENT

Compared with the preliminary version [1].

REFERENCES

[1] H. Dai, Y. Ji, L. Liu, G. Yang, and X. Yi, "A privacy-preserving multi-keyword ranked search over encrypted data in hybrid clouds," in *Proc. 5th Int. Conf. Artif. Intell. Secur. (ICAIS)*, New York, USA, 2019, pp. 68–80.
[2] S. Grzonkowski, P. M. Corcoran, and T. Coughlin, "Security analysis of authentication protocols for next-generation mobile and CE cloud services," in *Proc. IEEE Int. Conf. Consum. Electron.*, Berlin, Germany, Sep. 2011, pp. 83–87.
[3] R. L. Cilibrasi and P. M. Vitanyi, "The Google similarity distance," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 370–383, Mar. 2007.
[4] D. Xiaoding 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.
[5] N. Cao, M. Li, and W. J. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 829–839.

[6] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy–preserving multi–keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[7] J. Xu, W. Zhang, C. Yang, J. Xu, and N. Yu, "Two-step-ranking secure multi–keyword search over encrypted cloud data," in *Proc. Int. Conf. Cloud Service Comput.*, Shanghai, China, Nov. 2012, pp. 124–130.
[8] C. Yang, W. Zhang, J. Xu, J. Xu, and N. Yu, "A fast privacy–preserving multi-keyword search scheme on cloud data," in *Proc. Int. Conf. Cloud Service Comput.*, Shanghai, China, Nov. 2012, pp. 22–24.
[9] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling fine–grained multi–keyword search supporting classified sub–dictionaries over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 3, pp. 312–325, May 2016.
[10] H. Li, D. Liu, Y. Dai, and T. H. Luan, "Engineering searchable encryption of mobile cloud networks: When QoE meets QoP," *IEEE Wireless Commun.*, vol. 22, no. 4, pp. 74–80, Aug. 2015.
[11] 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, Feb. 2016.
[12] Z. Xiangyang, D. Hua, Y. Xun, Y. Geng, and L. Xiao, "MUSE: An efficient and accurate verifiable privacy-preserving multikeyword text search over encrypted cloud data," *Secur. Commun. Netw.*, vol. 2017, Nov. 2017, Art. no. 1923476.
[13] C. Chen, X. Zhu, P. Shen, J. Hu, S. Guo, Z. Tari, and A. Y. Zomaya, "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.
[14] 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 Conf. Comput. Commun.*, Piscataway, NJ, USA, May 2014, pp. 2112–2120.
[15] 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.
[16] 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, Aug. 2015.
[17] Z. Fu, X. Sun, N. Linge, and L. Zhou, "Achieving effective cloud search services: Multi-keyword ranked search over encrypted cloud data supporting synonym query," *IEEE Trans. Consum. Electron.*, vol. 60, no. 1, pp. 164–172, Feb. 2014.
[18] Z. Xia, Y. Zhu, X. Sun, and L. Chen, "Secure semantic expansion based search over encrypted cloud data supporting similarity ranking," *J. Cloud Comput.*, vol. 3, no. 1, Dec. 2014.
[19] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*. New York, NY, USA: Van Nostrand, 1994.
[20] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Providence, RI, USA, Jun. 2009, pp. 139–152.
[21] D. Liu and S. Wang, "Nonlinear order preserving index for encrypted database query in service cloud environments," *Concurrency Computat., Pract. Exper.*, vol. 25, no. 13, pp. 1967–1984, Sep. 2013.
[22] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order–preserving symmetric encryption," in *Advances in Cryptology—EUROCRYPT*. Heidelberg, Germany: Springer, Nov. 2009, pp. 224–241.
[23] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 13th Annu. ACM Symp. Theory Comput.*, Dallas, TX, USA, May 1998, pp. 604–613.
[24] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.
[25] Z. Fu, L. Xia, X. Sun, A. X. Liu, and G. Xie, "Semantic–aware searching over encrypted data for cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2359–2371, Sep. 2018.
[26] Y. Yang, J. Liu, S. Cai, and S. Yang, "Fast multi-keyword semantic ranked search in cloud computing," *Chin. J. Comput.*, vol. 40, pp. 158–171, Jun. 2017.
[27] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
[28] H. Delfs and H. Knebl, *UCI Machine Learning Repository*. New York, NY, USA: Springer, 2013.
[29] 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.
[30] Y. Zhuang, "Symmetric repositioning of bisecting K-means centers for increased reduction of distance calculations for big data clustering," in *Proc. IEEE Int. Conf. Big Data*, Washington, DC, USA, Dec. 2016, pp. 2709–2715.
[31] S. Chakraborty, N. K. Nagwani, and L. Dey, "Performance comparison of incremental k-means and incremental DBSCAN algorithms," *CoRR*, vol. abs/1406.4751, Jun. 2014.
[32] M. Bulut, "ReducedCBT and SuperCBT, two new and improved complete binary tree structures," *CoRR*, vol. abs/1401.7741, Jan. 2014.
[33] M. Lichman, *UCI Machine Learning Repository*. Irvine, CA, USA: Univ. California, 2013.
[34] Z. Wang and B. Meng, "A comparison of approaches to chinese word segmentation in hadoop," in *Proc. IEEE Int. Conf. Data Mining Workshop*, Shenzhen, China, Dec. 2014, pp. 844–850.

HUA DAI was born in 1982. He is currently an Associate Professor with the Nanjing University of Posts and Telecommunications and a member of CCF. His research interests include data management and security and database security.

YAN JI was born in 1995. She is currently pursuing the M.S. degree with the Nanjing University of Posts and Telecommunications. Her research interests include data management and security and cloud computing.

GENG YANG was born in 1961. He is currently a Professor and Ph.D. Supervisor with the Nanjing University of Posts and Telecommunications and a Senior Member of CCF. His research interests include cloud computing and security, data security, and privacy protection.

HAIPING HUANG was born in 1981. He is currently a Professor and Ph.D. Supervisor with the Nanjing University of Posts and Telecommunications. His current research interests are wireless sensor networks and information security.

XUN YI was born in 1967. He is currently a Professor and Ph.D. Supervisor with the Royal Melbourne Institute of Technology University. His research interests include information security and distributed data processing.