This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3047921, IEEE Transactions on Cloud Computing

Enhanced Semantic-Aware Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data

Xuelong Dai, Hua Dai, Chunming Rong, Geng Yang, Fu Xiao, Bin Xiao

***Abstract*—Traditional searchable encryption schemes based on the Term Frequency-Inverse Document Frequency (TF-IDF) model adopt the presence of keywords to measure the relevance of documents to queries, which ignores the latent semantic meanings that are concealed in the context. Latent Dirichlet Allocation (LDA) topic model can be utilized for modeling the semantics among texts to achieve semantic-aware multi-keyword search. However, the LDA topic model treats queries and documents from the perspective of topics, and the keywords information is ignored. In this paper, we propose a privacy-preserving searchable encryption scheme based on the LDA topic model and the query likelihood model. We extract the feature keywords from the document using the LDA-based Information Gain (IG) and Topic Frequency-Inverse Topic Frequency (TF-ITF) model. With feature keyword extraction and the query likelihood model, our scheme can achieve a more accurate semantic-aware keyword search. A special index tree is used to enhance search efficiency. The secure inner product operation is utilized to implement the privacy-preserving ranked search. The experiments on real-world datasets demonstrate the effectiveness of our scheme.**

***Index Terms*—semantic-aware search, searchable encryption, cloud computing, multi-keyword ranked search**

I. INTRODUCTION

RECENTLY, with the rapid development of cloud computing, an increasing number of users tend to outsource their local data to save on the cost of data storage and maintenance. However, cloud servers are able to access user data without constraints, i.e., the privacy of outsourced data should be protected. Thus, data owners choose to encrypt their data before outsourcing it to the cloud. It is therefore not merely a theoretical challenge but indeed a practical necessity to be able to apply effectively searchable encryption schemes to documents.

In recent years, many researchers have proposed a number of searchable encryption schemes in cloud computing, such as single keyword search [1]–[4], multi-keyword search [5]–[12], etc. Such a scheme typically uses the TF-IDF based vector space model and the secure kNN algorithm [13] to generate encrypted document vectors and encrypted indexes (inverted indexes, tree indexes, Bloom filters, etc.) and outsource them to the cloud. Afterwards, such a scheme generates a trapdoor for the queried keywords and returns the search result with searching the index according to the trapdoor. The TF-IDF model uses the appearance of the keyword to evaluate its importance in the document. However, these schemes do not consider the latent semantic features of the user's queried keywords and documents, which could lead to unsatisfactory search results [14].

The TF-IDF model is based on the bag-of-words (BoW) model [15] and is widely used for feature extraction. Besides the ignorance of semantic information, the vectors generated by the TF-IDF model are based on the entire keyword dictionary. Thus, such vectors are extremely high-dimensional and sparse. However, the TF-IDF model can provide a reasonable description of the importance of a keyword to a document, so in this paper we use the TF-IDF model as the basis of our keyword extraction method.

The LDA topic model is a semantic model that can be used for discovering latent semantics among texts. In this model, documents and keywords are viewed from the perspective of topics. A given keyword may be closely related to several topics while being irrelevant to others. A given document usually only discusses a few topics. Hence, the topic model can be used for semantic feature extraction. More specifically, for a particular set of documents, the topic model extracts three topics according to the model parameter. By analyzing the keywords most related to each topic, we can define the topics semantically, e.g., {computer, religion, sports}. Thus, we can infer the content of a document or a keyword based on its relation to such topics. In a multi-keyword search, we use the topic distribution of the user's queried keywords to represent the user's search intention. We obtain the most relevant documents according to the semantic relevance scores between the documents and the queried keywords.

Document retrieval is a popular topic in information retrieval (IR), and many related studies have been published [14], [16]–[20], etc. These approaches can achieve much better search accuracy than can the traditional TF-IDF model in ad hoc document retrieval. In this work, we first apply the widely used query likelihood model and propose a novel multi-keyword ranked search scheme over encrypted cloud data. The query likelihood model uses keyword information to perform searches; this approach can overcome the problem that the LDA topic model ignores the contribution of the queried keywords' occurrence to the search result. However, not all the keywords in the dictionary are equally important, and considering the entire dictionary is inefficient. Selecting the feature keywords from the dataset is a more efficient strategy. Inspired by the TF-IDF model and information gain, we can apply these evaluation metrics to the topic model for extracting

The corresponding author is H.Dai. X.Dai, H.Dai, G.Yang and F.Xiao are with College of Computer Science and Technology, Nanjing University of Posts and Telecommunications, China. Email:{xdai@ieee.org, daihua@njupt.edu.cn, yangg@njupt.edu.cn, xiaof@njupt.edu.cn}

C.Rong is with Center of IP-Based Services Innovation, University of Stavanger, Norway. Email:{chunming.rong@uis.no}

B.Xiao is with Department of Computing, Hong Kong Polytechnic University, China. Email:{csbxiao@comp.polyu.edu.hk}

the feature keyword dictionary. Because we propose a new search scheme based on a different language model, we first apply TREC robust 2004 dataset for accuracy evaluation. This dataset was not used in previous works, because most of them were based on the same measurement metric, TF-IDF score. Without modifying the relevance measurement metric, these schemes will get the same research results. This dataset also provides a standard metric for evaluating the search results. Thus, we use it as search accuracy measurement in this paper.

In this paper, we propose a semantic-aware multi-keyword ranked search scheme over encrypted cloud data based on the LDA topic model and the query likelihood model, which incorporates latent semantic features of the users' search intention in searchable encryption. The data owner first trains the LDA topic model with the document set, and the model generates the document-topic relevance matrix (DTR-matrix) and the keyword-topic relevance matrix (KTR-matrix). Each document has its topic vector that contains latent semantic features. Afterwards, the data owner extracts each document's feature keywords. The entire feature keyword dictionary and the query likelihood model are used to generate documents' keyword vectors. Next, the data owner merges the topic vector and the keyword vector into the document vector. Subsequently, the data owner uses the secure inner product operation to encrypt document vector and the document itself. After building the complete binary tree (CBTree) index with the encrypted document vector, the data owner outsources the index and the encrypted documents to the cloud server. When a multi-keyword ranked search is started to retrieve the k most relevant documents, the data user converts the queried keywords into a query vector and uses the same secure inner product operation to encrypt the vector as a trapdoor. Then, the trapdoor is sent to the cloud server as the query command. The cloud server uses the CBTree index to obtain k most relevant documents as the search result. Afterwards, encrypted documents with the top k relevance scores are returned to the data user. The proposed scheme outperforms the existing TF-IDF schemes in terms of search time and index space since the dimension of the topic vector is much smaller than the scale of the keyword dictionary (the dimension of a TF-IDF vector) in the existing schemes. In addition, the semantic precision and recall of the query results are measured to validate the accuracy of the proposed scheme.

Our paper's main contributions are summarized as follows.

- We apply the LDA topic model to searchable encryption over cloud data and enable sematic-aware privacy-preserving ranked searches. Document vectors and query trapdoors are built based on the LDA topic model. The lower-dimensional topic vector, used for relevance measurement, improves the efficiency of searches.
- We propose a feature keyword extraction method and for the first time apply the query likelihood model to searchable encryption over cloud data, which can be combined with the LDA topic model. The extracted feature keywords can improve the search performance of our scheme. We also propose a complete binary tree index and a greedy depth-first search algorithm to improve the search efficiency of our scheme.
- We perform a security analysis against the known ciphertext threat model and the known background threat model. Moreover, we perform experiments on two real-world datasets to test the search accuracy and efficiency of our scheme. The TREC 2004 robust retrieval track (TREC robust 2004) dataset is first used for search accuracy measurement of a ranked search scheme in cloud computing.

II. RELATED WORK

A. Privacy-preserving Keyword Search Schemes

The earlier studies focus on privacy-preserving single-keyword searchable encryption over cloud data. Song et al. [2] for the first time defined the problem model for the searchable encryption over outsourced cloud data and proposed a systematic searchable encryption for single keyword search. Wang et al. [3] used the effective TF-IDF model for feature extraction from documents and proposed a single keyword ranked privacy-preserving search scheme that applied the inverted index to fit the needs of different circumstances. Other schemes [1], [4] were also proposed for performing ranked search over encrypted data.

Multi-keyword search schemes have also been proposed to provide more usable services on the cloud server. Cao et al. [5] for the first time proposed a privacy-preserving multi-keyword ranked search scheme (MRSE) that used the secure kNN algorithm [13] to achieve multi-keyword searchable encryption. On the basis of MRSE, researchers used the tree-based index in searchable encryption to enhance the efficiency of various schemes. Sun et al. [6] proposed a secure multi-keyword search scheme with an MDB tree [21] as the search index, which improved search efficiency. The scheme achieved better than linear search efficiency but with precision loss. Xia et al. [8] adopted a special binary tree and used a "greedy depth-first search" algorithm to achieve sublinear search time. Their scheme also supported the dynamic update of documents and achieved sublinear search efficiency with accurate search result. We choose Xia's work as comparison scheme because it greatly improved the search performance of the multi-keyword ranked search scheme. Chen et al. [7] and Zhu et al. [9] adopted data-mining algorithms and proposed privacy-preserving ranked keyword search schemes. These schemes adopted different index methods to achieve more efficient ranked searches with tree-based indexes. However, in [7]'s work, the search result suffered from a precision loss. In [9], the scheme is complex to implement on a large dataset. In this work, we propose a complete binary tree-based index and achieve efficient and accurate semantic-aware multi-keyword ranked search.

B. Multi-keyword search schemes with semantic extensions

The feature extraction models used in traditional schemes such as the TF-IDF model ignore the hidden semantic information in keywords and the semantic relations between keywords. Thus, many schemes that support semantic search feature embedded semantic models. Considering synonyms'

expansion, Xia et al. [22] used the inverted index to implement multi-keyword searchable encryption. The scheme used the semantic information hidden in the queried keywords to perform searches with semantic extensions. Fu et al. [23] proposed a multi-keyword searchable encryption based on the MRSE scheme. The scheme entailed synonym expansion using a dictionary, resulting in a more accurate semantic search algorithm. With the implementation of stronger semantic tools, more semantic search models have been presented. Fu et al. [24] used a concept map to process the keywords and achieved the semantic search over encrypted cloud data. They also proposed a more efficient privacy-preserving multi-keyword search scheme based on a concept hierarchy [25] that supported semantic search with private and public servers. However, synonym expansion was based on the keywords extracted by the TF-IDF model which is not a semantic model. Accordingly, [22] and [23] attained smaller improvements in semantic search precision. [24] and [25] used stronger semantic tools that necessitated manually selecting the concept from the dictionary and could not be directly applied to a new dataset. Additionally, the schemes could be inefficient because the number of concepts can be more than the scale of the dictionary. Thus, we use a topic model to achieve accurate and efficient multi-keyword searches with a semantic extension.

The LDA topic model was first proposed by Blei et al. [26] and is an effective model for natural language processing. This model can also be used as a dimension reduction algorithm. Due to this model's effective semantic feature extraction, Heinrich et al. [27] proposed a text analysis scheme based on this model. Pang et al. [28] proposed a text search method for enterprise text searches by using the LDA topic model with topical intention obfuscation. Based on [28], Wang et al. [29] proposed a keyword search method that entailed masking the topical intent generated by the LDA topic model. In our previous work [30], we adopted the LDA topic model to achieve a multi-keyword ranked search scheme over encrypted cloud data. The topic information was used for supporting the semantic-aware search. Building on [30], we utilize topic information for keyword extraction and the query likelihood model to achieve a more accurate search, which solves "keywords ignorance" problem in our previous work. In addition, the combination of the topic vector and the keyword vector can further improve the security of our scheme, which will be discussed in the following sections.

III. NOTATIONS AND BACKGROUND KNOWLEDGE

A. Notation

- D — A set of n documents, $D = \{d_1, d_2, \ldots, d_n\}$.
- $\tilde{D}$ — A set of n encrypted documents, $\tilde{D} = \{\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_n\}$.
- T — A set of m topics, $T = \{t_1, t_2, \ldots, t_m\}$.
- W — A dictionary containing u keywords, $W = \{w_1, w_2, \ldots, w_u\}$.
- FW — A feature dictionary containing b feature keywords, $FW = \{fw_1, fw_2, \ldots, fw_b\}$.
- Q — A ranked search with multiple queried keywords, $Q = \{w_{p_1}, w_{p_2}, \ldots, w_{p_h}\}$.
- Θ — A $n\times m$-dimensional document-topic relevance matrix (DTR-matrix) that is the plaintext index.
- Θ_{new} — An $n\times(m+b)$-dimensional document-topic relevance matrix extended with feature dictionary FW.
- Ω —An $u\times m$-dimensional keyword-topic relevance matrix (KTR-matrix).
- V_d — A plaintext document vector.
- V_Q — A plaintext query vector.
- V_T — An m-dimensional topic probability vector.
- V_W — An u-dimensional keyword probability vector.
- $\tilde{V}_Q$ — A trapdoor vector that is the encrypted query vector of V_Q.

B. Multi-Keyword Ranked Search and Information Retrieval

Information Retrieval (IR) is a currently popular research topic, and many related works have been proposed [14], [16]–[20], etc (We only focus on ranked searches over encrypted cloud data, thus only few papers are discussed for clear and better understanding). However, ranked search schemes in cloud computing mostly use the old-fashioned TF-IDF model to perform searches over documents. The ad hoc document ranking in IR is similar to the aim of a multi-keyword ranked search. Three models are commonly used models in such ranking: BM25 [14], [16], query likelihood [17], [18] and neural ranking [19], [20] models. The main obstacle to directly using these models in a ranked search scheme over encrypted cloud data is that, for security reasons, the query must be encrypted before being uploaded to the cloud server. However, the ranking models need queried keywords to measure the relevance between documents and queries. Thus, it is difficult to adopt the state of art IR models in searchable encryption schemes.

We denote by $Score(Q, d)$ the relevance score between query Q and document d. In what follows, we briefly introduce the abovementioned models.

BM25 model:

$$Score(Q,d) = \sum_{i=1}^{h} IDF(w_{p_i}) \cdot \frac{f_{p_i}}{f_{p_i} + k_1 \cdot (1 - b + b \cdot \frac{dl}{avgdl})} \tag{1}$$

$$IDF(w_{p_i}) = \log \frac{(N - n(w_{p_i}) + 0.5)}{n(w_{p_i}) + 0.5} \tag{2}$$

where N is the total number of all documents, $n(w_{p_i})$ is the number of documents that contain queried keyword w_{p_i}, f_{p_i} is the number of occurrences of queried keyword w_{p_i} in the given document d, k_1 is a smoothing parameter, dl is the length of document d, and $avgdl$ is the average length of whole documents.

The BM25 model improves the performance of the traditional TF-IDF model and is more suitable for realistic applications. However, the BM25 model still only considers the occurrence of keywords and thus has the worst performance among the three models. It serves as the baseline of the ad hoc document retrieval task.

Query Likelihood model:

$$Score(Q,d) = P(Q|d) = \prod_{i=1}^{h} p(w_{p_i}|d) \tag{3}$$

$$p(w_{p_i}|d) = \frac{dl}{dl+\delta}P_{ML}(w_{p_i}|d) + (1 - \frac{dl}{dl+\delta})P_{ML}(w_{p_i}|coll) \quad (4)$$

where $P_{ML}(w_{p_i}|d)$ is the maximum likelihood estimate of word w_{p_i} in document d, $P_{ML}(w_{p_i}|coll)$ is the maximum likelihood estimate of word w_{p_i} in the entire collection, δ is the Dirichlet prior, and dl is the length of document d.

This is the basic structure of the query likelihood model with Dirichlet smoothing. The model adopts the likelihood of the given query as relevance measurement. It can be easily combined with probabilistic language models such as the LDA topic model to improve its precision. The query likelihood model performs better than the BM25 model [18]. In this paper, we first combine the query likelihood model with a multi-keyword ranked search scheme over encrypted cloud data.

Neural Ranking model:

The neural ranking model has the best performance among these there models, but it needs the neural ranking architecture to generate query representations that cannot be directly stored in the cloud server while manintaining both usability and security. Therefore, we do not further discuss the usage of the neural ranking model.

Integrating with the query likelihood model not only improves our scheme's ranked search accuracy but also enhances the security of our scheme, which will be discussed in the later sections. We also first use a standard IR dataset to evaluate the performance of the multi-keyword ranked search schemes over encrypted cloud data.

C. LDA Topic Model and Semantic-Aware Search

The Latent Dirichlet Allocation (LDA) model is presented by David Blei as a graphical model for topic discovery [26] that is the Bayesian version of the pLSA model. The LDA topic model uses an unsupervised learning algorithm and can be easily implemented in different circumstances. This model regards topics as the connections between documents and keywords. Every document is related to several topics, and every keyword has topics that are closely related to it. In the LDA topic model, the documents and keywords are only represented by their topic distributions, which improves both time and space efficiency of this model.

The document-topic relevance matrix (DTR-matrix) Θ and the keyword-topic relevance matrix (KTR-matrix) Ω generated by the LDA topic model are used in this paper. Because the topic is strongly defined semantically or epistemologically, we are able to perform semantic-aware searches with the topic relevance score.

Definition 1: Document Topic Vector. In the DTR-matrix Θ, the row vector $\Theta[i]$ is the document topic vector of document d_i, and $\Theta[i][j]$ represents the relevance score between d_i and topic t_j.

Definition 2: Keyword Topic Vector. In the KTR-matrix Ω, the row vector $\Omega[i]$ is the keyword topic vector of keyword w_i, and $\Omega[i][j]$ represents the relevance score between w_i and t_j.

The previous work in [30] proposed an efficient search scheme LDA-EMRSE, but the topic model ignored the information on keywords' presence in documents, and the documents returned as the ranking result by LDA-EMRSE might not contain the queried keywords. To overcome this disadvantage, we combine the topic model and the query likelihood model and propose a more accurate scheme.

D. Operation Definitions

Hadamard Product [31]. Given two n-dimensional vectors V_1 and V_2, the Hadamard product, denoted by $V_1 \circ V_2$, is an n-dimensional vector, and it is defined as

$$V_1 \circ V_2 =< V_1[1] \cdot V_2[1], V_1[2] \cdot V_2[2], \cdots, V_1[n] \cdot V_2[n] > \quad (5)$$

where $\circ$ is the Hadamard operator.

Secure Inner Product Operation. The Multi-keyword Ranked Search Scheme (MRSE) proposed in [5] uses the secure kNN algorithm [13] to implement the secure vector inner product. Given two n-dimensional vectors p and q, and secure key M in the form of an $n \times n$ random invertible matrix, p and q are converted to their encrypted forms $\tilde{p}$ and $\tilde{q}$, respectively, by matrix multiplication with M so that $\tilde{p} = M^T \cdot p$ and $\tilde{q} = M^{-1} \cdot q$. The inner product of p and q is calculated by their encrypted form $\tilde{p} \cdot \tilde{q}$ as (6) shows.

$$\begin{aligned} \tilde{p} \cdot \tilde{q} &= (M^T p)^T \cdot (M^{-1} q) \\ &= p^T M M^{-1} q \\ &= p \cdot q \end{aligned} \quad (6)$$

IV. PROBLEM FORMULATION

A. Ranked Search Model

In this paper, the ranked search model returns the ranked top-k documents from document set $D = \{d_1, d_2, \ldots, d_n\}$ with the k highest relevance scores. A ranked search is represented by a triple

$$Query = (D, Q, k) \quad (7)$$

where $Q = \{w_{p_1}, w_{p_2}, \ldots, w_{p_h}\}$ are the queried keywords and k is the number of requested documents, $k \ll n$. For simplicity, we use Q to represent a query.

Assuming that R is the ranked search result of a query Q, we have

$$|R| = k \wedge \forall d_i, d_j \left(d_i \in R \wedge d_j \in (D - R)\right) \rightarrow Score(d_i, Q) > Score(d_j, Q) \quad (8)$$

where *Score* is the measurement of relevance between the queried keywords and a document, which will be discussed in Section VI.

B. System Model

In this paper, we adopt the same system model as in [8], [32], which consists of three entities: the data owner, the data user and the cloud server. The detailed system model is shown in Fig. 2.

1) The data owner owns a document set $D = \{d_1, d_2, \ldots, d_n\}$. In our scheme, the data owner performs LDA training on D to generate the document-topic relevance matrix (DTR-matrix) Θ and the keyword-topic relevance matrix (KTR-matrix) Ω. Then, the data owner extracts the feature keywords

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3047921, IEEE Transactions on Cloud Computing

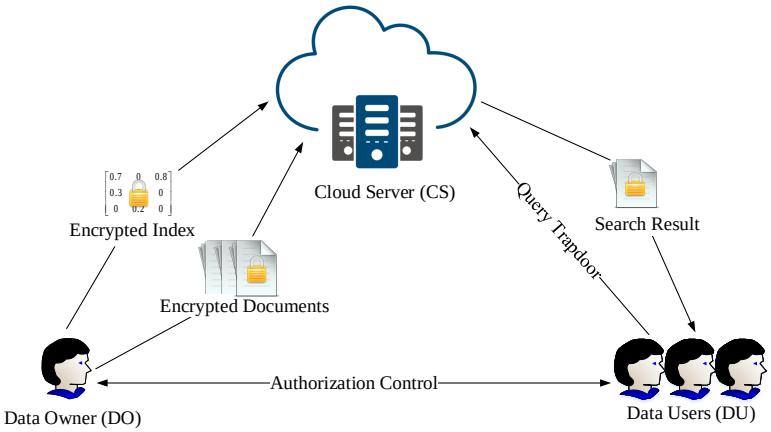


Fig. 1: System model.

in the dataset and generates the feature keyword vector for each document with feature dictionary FW. After that, the data owner merges the topic vector and the feature keyword vector to obtain the document vector V_d. Thus, Θ is extended to Θ_{new}, which is used to perform efficient and semantic-aware secured searches. After building the index CBTree $\mathcal{I}$ with the document vectors that are the rows of matrix Θ_{new}, the data owner encrypts the document set D and the index tree $\mathcal{I}$ into $\tilde{D}$ and $\tilde{\mathcal{I}}$. Finally, the data owner outsources $\tilde{\mathcal{I}}$ and $\tilde{D}$ to the cloud server.

2) The data user is the user authorized to perform searches on document set D. Assuming that the data user performs a ranked search with queried keywords $Q = \{w_{p_1}, w_{p_2}, \ldots, w_{p_h}\}$, the data user follows the search protocol and generates the trapdoor $\tilde{V}_Q$ of Q and then submits $\tilde{V}_Q$ to the cloud server. When the data user receives the search result in an encrypted form from the cloud server, the data user then decrypts the result with the secure key shared with the data owner to obtain the plaintext search result.

3) The cloud server provides computing and storage services to users. In the proposed scheme, the cloud server stores the encrypted index $\tilde{\mathcal{I}}$ and the encrypted document set $\tilde{D}$. Upon receiving the trapdoor $\tilde{V}_Q$ from the data user, the cloud server performs the privacy-preserving ranked search with $\tilde{\mathcal{I}}$ and $\tilde{V}_Q$. Having obtained the search result, the cloud server returns it to the data user and becomes ready for another request.

C. Threat Model

In the proposed scheme, we consider the cloud server to be "honest-but-curious", which is adopted in many related works [8], [33], [34], [5]. Specifically, the cloud server honestly and correctly implements operations in the presented protocols and returns the search result. However, the cloud server is "curious" about sensitive information for the sake of data inference and analysis. Given the information known to the cloud server, we mainly adopt the two threat models related to the privacy-preserving search scheme proposed by [5].

Known ciphertext threat model. In this model, the cloud server only knows the encrypted document set $\tilde{D}$, the encrypted CBTree index $\tilde{\mathcal{I}}$, and the search trapdoor $\tilde{V}_Q$. Thus, the cloud server could apply a ciphertext-only attack (COA) [35] in this model to obtain the plaintext data.

Known background threat model. In this model, the cloud server is able to acquire more knowledge, such as the term frequency (TF) statistics of the documents and the relation between the trapdoors. Using these types of information, the cloud server can infer or even recognize the retrieved keywords by using the trapdoor's information and by analyzing the histogram or the value range of the corresponding frequency distributions [34], [8].

D. Design Goals

Given a search request $Query = (D, Q, k)$, the proposed ranked search scheme aims at achieving the following goals:

Multi-keywords ranked search. The proposed scheme is designed to allow the cloud server to return k ranked and most semantically relevant ciphertext documents.

Search efficiency and effectiveness. The proposed scheme is designed to achieve efficient and accurate searches by using a complete binary tree-based index and a greedy depth-first search algorithm.

Privacy preservation. The proposed scheme is designed to protect sensitive information from the curious cloud server. Specifically, document confidentiality, index confidentiality, query confidentiality, trapdoor unlinkability, and keyword and topic privacy should be preserved. The specific requirements are as follows:

- *Document confidentiality*: the plaintext of the documents should be secured from the cloud server.
- *Index confidentiality and query confidentiality*: the CBTree $\mathcal{I}$, being the index of ranked searches, and queried keywords Q should be secured.
- *Trapdoor unlinkability* [8]: the cloud server should not be able to link similar search requests with the similar trapdoors or the similar search results.
- *Keyword and topic privacy*: the cloud server should not be able to identify the specific keywords using the topic information in trapdoors.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3047921, IEEE Transactions on Cloud Computing

V. FEATURE KEYWORD EXTRACTION AND QUERY LIKELIHOOD MODEL

The topic model is a powerful semantic model, and we achieve semantic-aware search by using the LDA topic model. However, after the topic model has been trained with document set D, the information of the topics rather than the keywords will represent the document. In other words, the search result returned by the topic model might not contain the queried keywords. Thus, in this paper, we consider that some of the queried keywords are important and that such keywords' information needs to be emphasized when searches are performed. To find the important (feature) keywords from D, we first describe the LDA-based information gain and the topic frequency-inverse topic frequency (TF-ITF) model. The former can measure the importance of a word in a topic set while the later can be used to evaluate the importance of a word in a document. Using these two models, we finally propose the feature keyword extraction method and the query likelihood model.

A. LDA-based Information Gain Model

Information gain [36] is defined as the difference in the total information entropy with the feature and without it, which is used for text feature selection. Information entropy is a measure of unpredictability or information content of a message source. Because we use topic information to select feature keywords, we calculate entropy by considering topic information as features. The binary entropy function $H(T)$ is defined in (9).

$$H(T) = -\sum_{i=1}^{|T|} P(t_i) \log P(t_i) \tag{9}$$

where t_i is the i^{th} topic in T, and $P(t_i)$ is the probability that topic i occurs in document set D. Given Θ, we can calculate $P(t_i)$ by the prior belief of topic i appearing throughout D.

The conditional entropy $H(T|w)$ measures the entropy of keyword w_i in T.

$$\begin{aligned} H(T|w_i) = -P(w_i) \sum_{j=1}^{|T|} P(t_j|w_i) \log P(t_j|w_i) \\ + P(\bar{w_i}) \sum_{j=1}^{|T|} P(t_j|\bar{w_i}) \log P(t_j|\bar{w_i}) \end{aligned} \tag{10}$$

where $P(w_i)$ is the probability that w_i appears in D. $P(t_j|w_i)$ is the probability that w_i belongs to topic t_j and is equal to $\Omega[i][j] \cdot P(t_j)$. Similarly, $P(t_j|\bar{w_i})$ is the probability that w_i does not belong to t_j and is equal to $(1 - \Omega[i][j]) \cdot (1 - P(t_j))$.

Thus, the information gain of w_i can be calculated by (11).

$$IG(w_i, T) = H(T) - H(T|w_i) \tag{11}$$

Information gain has been widely used for feature extraction in classification. A higher information gain means a greater importance of the word's contribution to the total information contained in the dataset. In this paper, we use a word's information gain to evaluate how well the word can describe the entire dataset semantically. Thus, we extract the feature keywords to represent the dataset according to their information gain.

B. Topic Frequency-Inverse Topic Frequency Model

We propose a new model called Topic Frequency-Inverse Topic Frequency (TF-ITF) that uses the topic information and the most widely used TF-IDF model. TF-ITF aims to find a more effective way to describe how words are semantically related to a specific document semantically. This model uses topic relevance rather than the number of keyword occurrences to measure the relevance to the document. The topic relevance can effectively describe the difference between words and documents. The detailed definitions are given as follows:

Definition 3: Topic Frequency. The topic frequency (TF) of keyword w_i is denoted by $tf(w_i)$. It measures the probability that keyword w_i is related to every topic t_j in topic set T.

$$tf(w_i) = \Omega[i] \tag{12}$$

where $\Omega[i]$ is the keyword topic vector of w_i, and $\Omega[i][j]$ represents the topic frequency of w_i in topic t_j.

The topic frequency of w_i and d_j is denoted by $tf(w_i, d_j)$. It measures the similarity between w_i and d_j from the perspective of topics. Instead of simply using the number of keyword occurrences, the topic frequency can evaluate the importance of keywords better from the perspective of semantics. It is calculated as the topic relevance between w_i and d_j,

$$\begin{aligned} tf(w_i, d_j) &= \prod_{k=1}^{|T|} P(w_i|t_k) \cdot P(t_k|d_j) \\ &= \Theta[j] \cdot \Omega[i] \\ &= \Theta[j] \cdot tf(w_i) \end{aligned} \tag{13}$$

where $\Theta[j]$ is the topic vector of d_j.

Similarly, we define the ITF value as the inverse topic frequency. Because the topic vector does not have zero values, we use a threshold ζ to filter the topics that are only marginally related to w_i.

Definition 4: Inverse Topic Frequency. Inverse topic frequency (ITF) of w_i is denoted by $itf(w_i)$. It measures the amount of information provided by w_i from the perspective of topics. ITF is defined in (14).

$$itf(w_i) = \log \frac{|T|}{1 + |t : t > \zeta|} \tag{14}$$

where t represents the topic frequency of w_i in each given topic.

Definition 5: Topic Frequency-Inverse Topic Frequency. The topic frequency-inverse topic frequency (TF-ITF) score between w_i and d_j is denoted by $tf\text{-}itf(w_i, d_j)$. It measures the importance of w_i in d_j.

$$tf\text{-}itf(w_i, d_j) = tf(w_i, d_j) \cdot itf(w_i) \tag{15}$$

The TF-ITF value considers both topic information and term frequency of words. The higher the TF-ITF value of w_i in d_j is, the higher the importance of w_i in d_j.

C. Feature Keyword Extraction

Using the TF-ITF score and information gain, we define the keyword importance score as shown in (16). For simplicity, we use *tfitf* to represent the final score.

$$tfitf(w_i, d_j) = tf\text{-}itf(w_i, d_j) \cdot IG(w_i, T) \tag{16}$$

where *tf-itf* is the TF-ITF score of keyword w_i, and $IG(w_i, T)$ is that keyword's information gain.

Because $itf(w_i)$ only considers the keyword's number of occurrences among topics, we add information gain to evaluate the difference between topics and each keyword's contribution to topics more accurately.

The *tfitf* score represents the keyword's importance in a specific document. We can extract the feature keywords from documents by selecting keywords with the higher *tfitf* scores. Using the keywords with higher *IG* or *tfitf* scores, we can construct the feature dictionary FW from document set D by performing the following three steps.

First, we calculate all keywords' values of information gain in dictionary W and select the top κ keywords to build FW_{IG} as a part of FW. The words in FW_{IG} contain more information than do others.

$$|FW_{IG}| = \kappa \wedge \forall w_i \in FW_{IG} \rightarrow IG(w_i, T) \geq IG(w_j, T), w_j \in W - FW_{IG} \tag{17}$$

Second, we calculate the *tfitf* scores of every keyword in document d_j and select the top λ keywords to build FW_{tfitf,d_j}. The keywords in FW_{tfitf,d_j} are the most suitable words that can represent d_j's topic information. Then, $FW_{tfitf,D}$ is built by merging every FW_{tfitf,d_j}.

$$\left|FW_{tfitf,d_j}\right| = \lambda \wedge \forall w_i \in FW_{tfitf,d_j} \rightarrow tfitf(w_i, d_j) \geq tfitf(w_j, d_j), w_j \in W - FW_{tfitf,d_j} \tag{18}$$

Third, the feature dictionary FW is built by merging FW_{IG} and $FW_{tfitf,D}$.

$$FW = FW_{IG} \cup FW_{tfitf,D} \tag{19}$$

Using the feature dictionary FW can improve search performance if queried keywords are important and meaningful and can overcome the drawback that the topic model in LDA-MRSE [30] ignores the specific keywords in searches.

D. Query Likelihood Model

After feature keyword extraction, we need to evaluate the relevance between each feature keyword and documents. In ad hoc document retrieval, the basic approach to using language models for IR involves a query likelihood model. Thus, we adopt the query likelihood model of [18] to score each feature keyword in each document as (20) shows.

$$p(fw|d) = \gamma(\frac{|d|}{|d|+\delta}P_{ML}(fw|d) + (1 - \frac{|d|}{|d|+\delta})P_{ML}(fw|D)) + (1-\gamma)P_{LDA}(fw|d) \tag{20}$$

where fw is a feature keyword from FW, d is a document, and both γ and δ are Dirichlet smoothing parameters.

The maximum likelihood estimate of fw is calculated by

$$P_{ML}(fw|d) = \frac{n(fw)}{|d|} \tag{21}$$

where $n(fw)$ is the number of fw's occurrences in d. Furthermore,

$$P_{ML}(fw|D) = \frac{N(fw)}{|D|} \tag{22}$$

where $N(fw)$ is the number of fw's occurrences in D.

In original method of [18], the researchers calculate $P_{LDA}(fw|d)$ by performing the Gibbs sampling from the LDA model. However, our experiments show that using $tf(w_i, d_j)$ performs slightly better. Thus, we calculate P_{LDA} by

$$P_{LDA}(fw|d) = tf(fw, d) \tag{23}$$

The value of $p(fw|d)$ represents both the keyword occurrence information and the LDA topic information that can be used to implement semantic-aware multi-keyword ranked search. If the feature keyword does not appear in d, it is still related to d according to $(1 - \frac{|d|}{|d|+\delta})P_{ML}(fw|D))$; however, we set $P_{LDA}(fw|d)$ to 0 because the Gibbs sampling is only performed on the keywords of d.

The query likelihood score between a feature keyword fw and a document d is defined as

$$QLScore(d, fw) = \begin{cases} p(fw|d) & fw \in d \\ (1 - \frac{|d|}{|d|+\delta})P_{ML}(fw|D) & fw \notin d \end{cases} \tag{24}$$

In the original query likelihood model, the model scores the query and the document according to (3). Due to security concerns, we cannot send the queried keywords directly to the cloud server; however, we can store the relevance score in advance with the feature dictionary to create the query likelihood model for searchable encryption using the logarithm of the respective value.

$$QLScore(d, Q) = V_Q^{FW} \cdot V_d^{FW} = \sum_{k=1}^{|FW|} V_Q^{FW}[k] \cdot V_d^{FW}[k] = \sum_{i=1}^{|Q|} \log(p(fw_i|d)) = \log \prod_{i=1}^{|Q|} p(fw_i|d) = \log p(Q|d) \tag{25}$$

where V_Q^{FW} and V_d^{FW} are the query feature keyword vector and the document feature keyword vector, respectively.

VI. ENHANCED SEMANTIC SEARCH MODEL

With the implementation of the LDA topic model, we propose the enhanced multi-keyword semantic ranked search model for a searchable encryption scheme. To overcome the problem of queried keywords being ignored in our previous work [30], we use both topic relevance and keywords' *QLScore* to obtain the search result in the enhanced search model. The latter contains two parts: Build Index and Perform Search. The detailed procedures are as follows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3047921, IEEE Transactions on Cloud Computing

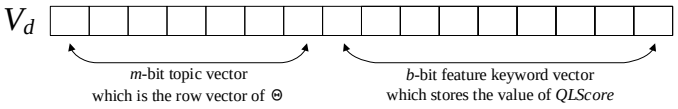


Fig. 2: Structure of V_d.

A. Build Index

The LDA topic model is used to extract semantic meanings from the document set and adopts topic information to represent documents and keywords. Each topic is related to keywords with the topic relevance score. The keywords with high relevance scores of a given topic are semantically related. However, a keyword could have different meanings in a different contexts, and could thus be related to different keywords in different topics. However, the topic itself does not have specific semantic meanings. Documents' topic relevance scores can also be used for relevance measurement.

The feature dictionary FW is used for better search accuracy. Not all keywords in the document set are equally important and taking the whole keyword dictionary into consideration wastes storage space and computing resources because the topic relevance is the major measurement. As mentioned in Section V, we use the IG score to evaluate keywords' importance in the dataset and the *tfitf* score to evaluate keywords' importance in each document with the topic model. Using the *tfitf* and IG scores, we build the feature dictionary FW. Finally, we use the query likelihood model to measure the relevance scores between feature keywords and documents to improve search accuracy.

First, the search model uses the LDA topic model to generate the DTR-matrix Θ and the KTR-matrix Ω after the completion of training with the document set D.

Second, the model calculates each keyword's IG score and the *tfitf* scores of each document's keywords. Hence, the feature dictionary is built according to the steps described in Section V.C. Afterwards, we use the query likelihood model to calculate each feature keyword's relevance to each document. Then, the model extends the original DTR-matrix Θ with feature dictionary FW that stores the values of $QLScore$ between documents and feature keywords. We assume that there are m topics, and b feature keywords in FW, and the extended DTR-matrix Θ_{new} is then an $n \times (m+b)$ dimensional matrix, and is utilized as the index. The structure of the extended document vector V_d is shown in Fig. 2.

B. Perform Search

The search model generates the query topic vector with Θ, Ω and FW, and performs a search with the query topic vector and the search index. Because of the property of the Dirichlet prior, we use the prior beliefs of keywords to generate the query topic vector. Before a search is performed, the model needs to generate two vectors:

The topic probability vector V_T records the prior belief of the corresponding topic in T appearing in D. Assuming that each document is equally important, the prior belief of topic t_i is the average probability that each document in D belongs to topic t_i, i.e.,

$$V_T = \frac{1}{n} \cdot \sum_{j=1}^{n} \Theta[j] \tag{26}$$

where $\Theta[j]$ is the j^{th} row vector of Θ.

The keyword probability vector V_W records the prior belief of each keyword in dictionary W appearing in D. We calculate the keyword probability vector using the prior belief of topics and the topic distribution of keywords, as shown in (27).

$$V_W = V_T \cdot \Omega^T \tag{27}$$

V_T and V_W are calculated only once and can be utilized for all queries. Afterwards, the model is ready for performing searches. The following four steps are involved in performing a query $Q = \{w_{p_1}, w_{p_2}, \ldots, w_{p_h}\}$.

First, the model generates the query topic vector T_Q for Q with h queried keywords. It is an m-dimensional vector and represents the topic intention of the queried keywords. The calculation of T_Q is shown in (28).

$$T_Q = \sum_{w_{p_i} \in Q} \frac{V_T \circ (\Omega[p_i] - \min(\Omega[p_i]))}{|Q| \cdot V_W[p_i]} \tag{28}$$

where w_{p_i} represents the ${p_i}^{th}$ keyword in keyword dictionary W.

Second, the model extends the query topic vector with feature dictionary FW. We use 1 or 0 to represent that a feature keyword does or does not appear among the queried keywords. Thus, the extended query vector V_Q is now an $(m + b)$-dimensional vector.

Third, the model measures the relevance scores between the documents and the query by calculating the inner products of vectors that correspond to them. Given a document d, the relevance score between Q and d is computed as shown in (29).

$$\begin{aligned} Score(d, Q) = V_Q \cdot V_d &= \sum_{k=1}^{|T|+|FW|} V_Q[k] \cdot V_d[k] \\ &= \alpha TRScore(Q, d) + \beta \sum_{w_i \in FW} QLScore(d, w_i) \end{aligned} \tag{29}$$

where V_Q and V_d are the query vector and the document vector, respectively. $TRScore(Q, d) = \sum_{k=1}^{|T|} V_Q[k] \cdot V_d[k]$ is the topic relevance between Q and d, which means the semantic relevance. The $QLScore$ represents the feature keyword relevance based on the query likelihood model. α and β are weight parameters to balance the importance of semantic relevance and feature keywords relevance. Users can modify α and β to get a satisfying search result (this will be discussed in Section VIII). The calculation of relevance score is the basic operation of ranked searches.

Fourth, as we discussed in Section IV.A, using the relevance measurement $Score(d, Q)$, we can obtain the top-k search result R of search request Q as (we reuse (8) for convenience):

$$|R| = k \wedge \forall d_i, d_j \left(d_i \in R \wedge d_j \in (D - R)\right) \rightarrow Score(d_i, Q) > Score(d_j, Q) \quad (30)$$

The documents in R are k documents that have higher relevance scores with Q among all documents in D.

VII. ENHANCED SEMANTIC SEARCH SCHEME

We implement the enhanced semantic search scheme (LDA-ESSS) by adopting the enhanced semantic search model mentioned in Section VI. We use MRSE [5] as the basic framework. To provide efficient multi-keyword searches, we utilize a complete binary tree (CBTree) in LDA-ESSS. Specifically, the proposed scheme consists of two modules: the Setup Module and the Search Module. The detailed descriptions of the algorithms are provided in the following subsections. We first describe the build and search algorithms for CBTree.

A. Algorithms for CBTree

For efficiency, we use a complete binary tree (CBTree) as the search index in our scheme. Each node $\mathcal{I}[i]$ in CBTree has the same data structure, which is denoted as

$$\mathcal{I}[i] =< docID, d_v, fv > \quad (31)$$

where $docID$ represents the corresponding document d_i's ID i, d_v denotes d_i's document vector, and fv denotes the filter vector of $\mathcal{I}[i]$ that stores the largest value of each dimension of the document vectors among all the child nodes.

We use an array structure $\mathcal{I}[1, 2, \ldots, n]$ to store the entire CBTree according to [37]. Each element $\mathcal{I}[i]$ in $\mathcal{I}$ represents a tree node in CBTree. Then

- Each node $\mathcal{I}[i]$ sets its $docID$ to i and d_v to d_i's document vector. The root node is $\mathcal{I}[1]$.
- If $\mathcal{I}[i]$ has the left and right child nodes, they are $\mathcal{I}[2i]$ and $\mathcal{I}[2i+1]$, respectively. Its filter vector fv is calculated by $fv[j] = \max\{\mathcal{I}[2i].fv[j], \mathcal{I}[2i+1].fv[j], d_v[j]\}$, where $j = 1, 2, \ldots, |d_v|$.
- If $\mathcal{I}[i]$ is not the root, its parent node is $\mathcal{I}\lfloor i/2 \rfloor$. Its filter vector $fv = d_v$.

Because every node stores a document vector, we construct the CBTree with n nodes in total. To build a CBTree, we first create a complete binary tree with n nodes. Then, we calculate each node's filter vector fv from bottom-up. The tree construction algorithm is shown in [30] and denoted by *BuildCBTree* in this paper. Additionally, the complete binary tree is the shortest tree among all binary trees. The height of the CBTree will be $\lceil \log_2(n+1) \rceil$. Thus, we can achieve a sublinear search complexity with the CBTree, and the structure of our tree further improves the search efficiency.

The search process follows a greedy depth-first search algorithm that recursively traverses the nodes, starting from the root node. We use the search result list R to store the top-k documents with the largest relevance scores. The relevance score calculation is described in Section VI and denoted by function $Score()$. The last document in R, denoted by $R[k]$, has the lowest score among all documents in R and is used to speed up the search process. The relevance score between the filter vector in each node $\mathcal{I}[i]$ and the query vector is the largest possible score among all nodes in the subtree with the root as node $\mathcal{I}[i]$. Using this score, we can reduce the calculation time by filtering these nodes. In addition, we need to keep R ranked in the descending order and sorted when a new document is added. The detailed search algorithm is shown in [30]. Given a search request Q, the search process starts with $\mathcal{I}[1]$, and it is briefly described as follows (an example is given in Fig. 3).

1) The algorithm first calculates $Score(\mathcal{I}[i].fv, Q)$; if $Score(\mathcal{I}[i].fv, Q) < Score(R[k], Q)$, then it will not continue searching in this subtree.
2) The algorithm calculates $Score(\mathcal{I}[i].d_v, Q)$; if $Score(\mathcal{I}[i].d_v, Q) > Score(R[k], Q)$, then it adds the corresponding d_i into R.
3) If node $\mathcal{I}[i]$ has child nodes, the algorithm traverses its child nodes $\mathcal{I}[2i]$ and $\mathcal{I}[2i+1]$. It first traverses the child node with a higher relevance score between Q and fv, and then the other node.

B. Algorithms in Setup Module

To achieve the semantic search over encrypted cloud data, the data owner needs to generate each document's vector that is merged by its topic vector and feature keyword vector. Then, the data owner builds the search index based on the CBTree, and outsources the data to the cloud server and the data user. The detailed algorithms are as follows.

GenKey$(\mathbf{1}^{l(n)})$: The data owner generates the secured key $SK = \{S, M_1, M_2, g\}$, where S is a random $(m+b)$-bit split vector, M_1 and M_2 are both random $(m+b)\times(m+b)$ invertible matrices, and g is used for document encryption. SK is only shared with the data owner and the data user.

BuildIndex(D): The data owner uses the LDA topic model to generate the DTR-matrix Θ and the KTR-matrix Ω with Gibbs sampling [26]. Then, the data owner calculates each keyword's IG score and selects the top κ words as feature keywords. Additionally, the data owner chooses the keywords with top λ highest *tfitf* scores as feature keywords for each document d_i in D. The feature dictionary FW is constructed for document set D. Afterwards, the data owner uses the query likelihood model to calculate the feature keyword relevance $QLScore$. The document vector results from merging by the topic vector and the feature keyword vector that stores the $QLScore$. Thus, Θ is extended to Θ_{new}, an $n \times (m+b)$ matrix, and each row of Θ_{new} is the document vector of a document. The specified procedure is illustrated in Section VI.A. Finally, the data owner executes the $BuildCBTree$ algorithm to generate the plaintext index tree $\mathcal{I}$ for D.

EncData$(SK, \mathcal{I}, D)$: The data owner encrypts $\mathcal{I}$ and D to the corresponding encrypted forms $\tilde{\mathcal{I}}$ and $\tilde{D}$. The procedures are as follows.

1) The data owner encrypts $\mathcal{I}$ with SK, and divides both the document vector and the filter vector of each node in $\mathcal{I}$ into two random vectors with S. Then the data owner encrypts them by the secure kNN algorithm [13]. Specifically, two random

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3047921, IEEE Transactions on Cloud Computing

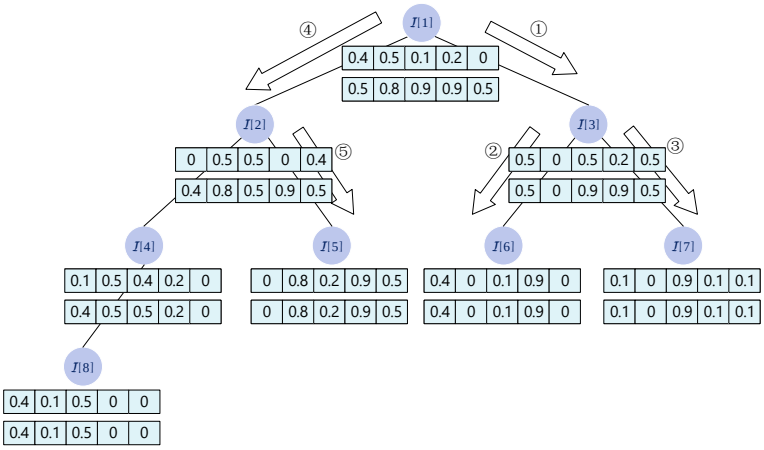


Fig. 3: Example of the CBTree-based search algorithm. The entire dataset contains 8 documents $D = \{d_1, d_2, \ldots, d_8\}$. We set the number of topics to 3 and extract two feature keywords. The document vector and the filter vector are the first row and the second row of each node $\mathcal{I}[i]$. Each vector is a 5-*bit* vector: the first three dimensions are the topic vector, and the last two dimensions are the feature keyword vector. Each tree node $\mathcal{I}[i]$ represents a corresponding document d_i. Given a search request Q with query vector $V_Q = \{0.9, 0.1, 0, 0.9, 0\}$ and k=3 that signifies that we want three most relevant documents, the search starts at the root node $\mathcal{I}[1]$, and the algorithm directly adds d_1 into result list R with the relevance score of 0.59. Then, it reaches $\mathcal{I}[3]$ because the latter has a higher relevance score of 1.26, and d_3 is added into R with the score of 0.63. Afterwards, d_6 is added into R with the score of 1.17 and d_7 is ignored because its score of 0.18 is less than the lowest relevance score of 0.59 in R. Next, the algorithm reaches $\mathcal{I}[2]$ and then $\mathcal{I}[5]$. Because d_5 has a relevance score of 0.89, it replaces d_1 in R. Finally, the algorithm reaches completion because the filter vector of $\mathcal{I}[4]$ has a lower relevance score of 0.59 than 0.63 in R.

vectors are generated according to the following procedure, where *Rand*() is a pseudo random generator. For simplicity, we use V to represent both the document vector and the filter vector.

- If $S[i] = 1$, then $V_1[i] = Rand()$ and $V_2[i] = V[i] - V_1[i]$.
- If $S[i] = 0$, then $V_1[i] = V_2[i] = V[i]$.

2) The data owner encrypts $\{V_1, V_2\}$ to $\tilde{V} = \{\tilde{V}_1, \tilde{V}_2\}$ by using the secure matrices M_1 and M_2 in SK.

$$\tilde{V} = \{\tilde{V}_1, \tilde{V}_2\} = \{M_1^T \cdot V_1, M_2^T \cdot V_2\} \tag{32}$$

Thus, the secure search index $\tilde{\mathcal{I}}$ is generated.

3) The data owner encrypts every document in D with the key g in SK and generates the set of encrypted document set $\tilde{D}$.

Finally, the data owner outsources $\tilde{D}$ and $\tilde{\mathcal{I}}$ to the cloud server and shares SK and Ω with the data user. The setup module's execution is finished, and the system is ready to perform ranked searches.

C. Algorithms in the Search Module

GenTrapdoor(Q, SK): The data user generates the query trapdoor $\tilde{V}_Q = \{\tilde{V}_{Q1}, \tilde{V}_{Q2}\}$ with a search request $Q = \{w_{p_1}, w_{p_2}, \ldots, w_{p_h}\}$, where w_{pi} represents the $p_i{}^{th}$ keyword in W. The detailed steps are as follows.

1) The data user generates the query vector V_Q for Q by following the procedure mentioned in Section VI.B. For a specific dataset, the data user can use parameters α and β to adjust the importance of topic relevance and keyword relevance. Thus, the data user multiplies the query's topic vector T_Q by α, obtaining the first m dimensions of V_Q and the query's feature keyword vector by β, obtaining the last b dimensions of V_Q. Accordingly, the query vector V_Q is generated (α and β can affect the ranked search result and reflect users' different search intentions).

2) The data user uses S to divide V_Q into two vectors $\{V_{Q1}, V_{Q2}\}$. For the i^{th} dimension of V_Q, the procedures follow two rules:

- If $S[i] = 0$, then $V_{Q1}[i] = Rand()$ and $V_{Q2}[i] = V_Q[i] - V_{Q1}[i]$.
- If $S[i] = 1$, then $V_{Q1}[i] = V_{Q2}[i] = V_Q[i]$.

3) The data user uses the secure matrices M_1 and M_2 in SK to encrypt $\{V_{Q1}, V_{Q2}\}$ into $\tilde{V}_Q = \{\tilde{V}_{Q1}, \tilde{V}_{Q2}\}$, as shown in (33).

$$\tilde{V}_Q = \{\tilde{V}_{Q1}, \tilde{V}_{Q2}\} = \{M_1^{-1} \cdot V_{Q1}, M_2^{-1} \cdot V_{Q2}\} \tag{33}$$

LDASearch$(\tilde{\mathcal{I}}, \tilde{V}_Q, R, k)$: The cloud server executes the greedy depth-first algorithm described in Section VII.A with $\tilde{\mathcal{I}}$ and obtains the search result list R. Using to the secure inner product operation (34), the semantic relevance score between the query vector and the document vector or the filter vector can be calculated from their encrypted forms. Then, the cloud server returns R to the data user.

$$\begin{aligned} Score(\tilde{V}, \tilde{V}_Q) &= \{M_1^T V_1, M_2^T V_2\} \cdot \{M_1^{-1} V_{Q1}, M_2^{-1} V_{Q2}\} \\ &= (M_1^T V_1)^T \cdot M_1^{-1} V_{Q1} + (M_2^T V_2)^T \cdot M_2^{-1} V_{Q2} \\ &= V_1 \cdot V_{Q1} + V_1 \cdot V_{Q2} \\ &= V \cdot V_Q \\ &= Score(V, V_Q) \end{aligned} \tag{34}$$

After receiving R, the data user decrypts the encrypted documents with the secure key g, shared by the data owner, and obtains the plaintext ranked search result.

D. Security Analysis

We analyze the privacy of our scheme according to the requirements that the privacy of documents, index (CBTree) and queried keywords, as well as trapdoor unlinkability and the keyword information be protected from the curious cloud server. We present proofs in what follows.

Theorem 1: The privacy of documents is protected in LDA-ESSS.

Proof: In LDA-ESSS, the documents themselves are not involved in the search process and are encrypted by well-established symmetric encryption (such as AES). Since SK is kept from the cloud server, the privacy of documents is preserved, and LDA-ESSS is CPA-secure for documents. ■

Theorem 2: The privacy of the CBTree and the queried keywords in Q are protected from the curious cloud server in LDA-ESSS.

Proof: In our scheme, $\mathcal{I}$ and V_Q are encrypted with secure key SK, and the cloud server cannot directly get the plaintext vectors from their encrypted forms. The secure matrices M_1 and M_2 are both randomly generated. Because we use the secure kNN algorithm, according to [8], M_1 and M_2 have been secured from the cloud server. For any vector $V = \{V_1, V_2\}$ in $\mathcal{I}$, it is encrypted and then outsourced to the cloud server. The encryption of V is:

$$\begin{cases} V_1 \cdot M_1^T = \tilde{V}_1 \\ V_2 \cdot M_2^T = \tilde{V}_2 \end{cases} \tag{35}$$

In (35), this quadratic equation only has $2n \times (m + b)$ known numbers, but need to solve $2n \times (m + b) + (m + b) \times (m + b)$ unknown numbers. Thus, the cloud server cannot achieve decryption and obtain the plaintext of $\mathcal{I}$ only with the ciphertext $\tilde{\mathcal{I}}$. The encryption of Q is the same as $\mathcal{I}$, thus the privacy of the CBTree and the queried keywords in Q are secured. ■

Theorem 3: Trapdoor unlinkability is preserved in LDA-ESSS.

Proof: According to the algorithm ***GenTrapdoor***(), the query trapdoor is split by a random split vector S and then encrypted by the secure kNN algorithm. Specifilly, for a query Q having multiple query keywords, the corresponding query vector V_Q is processed by two rules: if $S[i] = 0$, then $V_{Q1}[i] = Rand()$ and $V_{Q2}[i] = V_Q[i] - V_{Q1}[i]$; if $S[i] = 1$, then $V_{Q1}[i] = V_{Q2}[i] = V_Q[i]$. After that, the generated $V_Q = \{V_{Q1}, V_{Q2}\}$ is encrypted to form the query trapdoor.

For the same search request with the same queried keywords, the data user will generate different query trapdoors, because of the pseudo random generator $Rand()$ in the trapdoor generation algorithm. The cloud server cannot link the query trapdoors even if they are generated by the same queried keywords. Thus, trapdoor unlinkability is guaranteed.

However, the cloud server can link instances of the same search request with the same relevance scores between documents and the trapdoor, and the access patterns of the CBTree. With this information, the cloud server can link the same queries without acknowledging the linkability between the trapdoors. The reason is that despite different query trapdoors being generated, the relevance scores between the documents and the same queries are the same, as are the ranked results. To prevent linkability of queries, a better scheme needs to change the ranking and relevance scores, which will surely reduce the search accuracy. In our scheme, the data user can easily adjust α and β to obtain different rankings and relevance scores without modifying the encrypted data. Additionally, α and β can satisfy the user's different search intentions. Hence, the cloud server cannot infer the occurrences of the same search requests with the same queried keywords because the ranked results will be different. The detailed discussion about rank privacy is given in the experiment section. ■

Theorem 4: The privacy of keywords is preserved in LDA-ESSS.

Proof: As we discussed in [30], the topic vector replaces the original TF vector of a document in the topic model; thus, the privacy of keywords is preserved. The topic itself does not have specific meanings, and the cloud server can merely analyze the keywords based on the topic's statistical information. Additionally, the query trapdoor's topic distribution generated by the data user is calculated by all the queried keyword. For different queried keywords, the query's topic distribution could be totally different. Thus, the cloud server cannot infer the keywords based on the topic information.

However, in LDA-ESSS we extend the original topic vector with the feature keyword vector, and the feature keyword information should also be preserved. The feature keyword relevance score can obfuscate the topic relevance score, and vice versa. Thus, the privacy of keywords is enhanced in LDA-ESSS. Additionally, because of α and β, the ranked results will be further obfuscated. The data user will get more semantic search results with higher α and will get more queried keywords-related search results with higher β. A change in either parameter will change the relevance scores between the query and documents. Thus, the privacy of keywords is preserved in LDA-ESSS. We can also add phantom terms to enhance privacy according to [8], but it can affect the accuracy of our scheme. ■

In the last part of the security analysis, we want to clearly illustrate the security differences between LDA-ESSS and traditional TF-IDF based schemes.

The security requirements above have been widely adopted in the TF-IDF based schemes. The proposed LDA-ESSS and the TF-IDF based schemes both adopt the secure kNN algorithm to achieve searchable encryption, and this algorithm is the most widely adopted encryption algorithm that can achieve multi-keyword ranked search scheme over encrypted cloud data. However, in the TF-IDF based schemes, a curious cloud server can easily link similar search requests. The reason these schemes suffer from query linkability problems is that the TF-IDF model uses the sum of queried keywords' $TF{\times}IDF$ values as the relevance score. For two queries $Q_1 = \{w_1, w_2\}$ and $Q_2 = \{w_2\}$ and a document d, if w_2 appears in d while w_1 does not. The relevance score $Score(Q_1, d)$ will be the same as $Score(Q_2, d)$ because $TF \times IDF$ between w_1 and d is 0, and this will causes a leakage of keyword information. Even with randomization, the cloud server will still be easily able to deduce the queried keywords using the relevance scores' distribution because the rank scores are linearly related to the keywords. Furthermore, randomization will decrease the

ranking's accuracy.

In contrast, in LDA-ESSS, we use topic information instead of the $TF \times IDF$ values to represent documents and keywords. As (29) shows, the topic relevance score is calculated from the query's and document's topic representations. Considering the example above, $Score(Q_1, d)$ will be different from $Score(Q_2, d)$ because T_Q is generated based on the topic prior beliefs of queried keywords. A feature keyword's relevance is calculated by multiplication with each queried keyword's likelihood despite its occurrence in the document. There is no direct linear relationship between the queried keywords and relevance scores in both topic relevance and feature keyword relevance. Thus, little information can be acquired by the cloud server, and our scheme is more secure than the TF-IDF based schemes.

As to the leakage of rank results, the document rankings resulting from a given set of queried keywords will be the same. Using different values of α and β can change the rank results, but accuracy will be affected.

The main difference in information leakage between TF-IDF based scheme and LDA-ESSS is that our scheme only leaks topic information, while TF-IDF based schemes directly leak keyword information. The documents and keywords have different topic distributions, so the cloud server cannot directly infer whether a queried keyword appears in a specified document. This makes LDA-ESSS a more secure ranked search scheme.

VIII. PERFORMANCE ANALYSIS

We implement our scheme using Java 14.0.2 on a PC with an Intel Core(TM) I5-8400 CPU running Windows 10. We test our scheme by using two different real-world datasets: 20 newsgroups [38] and the TREC robust 2004 [39] dataset. The 20 newsgroups dataset is used for semantic accuracy evaluation. It is a categorized dataset of 20 different categories of news that contains 11000 articles in total. We choose different categories and evaluate the performance of our scheme in different settings. To make our assessment more convincing, we compare our scheme with two different schemes, namely, LDA-MRSE [30] and TFIDF-MRSE [8]. The TREC robust 2004 dataset is widely-used IR document retrieval. However, there is no related studies of multi-keyword ranked search schemes have used the TREC robust 2004 dataset as a test dataset. Because these schemes that adopted the TF-IDF model will get similar search result in this dataset. Thus, for the first time, we use it as our experiment dataset for keyword search accuracy evaluation over encrypted data. This dataset has 500k documents in total. We compare our scheme with TFIDF-MRSE [8] and BM25-MRSE.

A. Parameters

We measure the semantic accuracy on the 20 newsgroups dataset and the keyword accuracy on the TREC robust 2004 dataset. Thus, parameter settings are different.

Parameters for the 20 newsgroups dataset are shown in Table I: n, m and k are the numbers of documents, topics and required documents, α and β are thresholds used for search, ζ, κ and λ are thresholds used for keyword extraction, and γ and δ are the Dirichlet priors for the query likelihood calculation, set according to [18] to $\gamma = 0.7$ and $\delta = 1000$.

TABLE I: Default Parameters for the 20 newsgroups dataset

Parameter	n	m	k	α	β	ζ	κ	λ
Value	8000	70	200	50.0	1.0	0.005	1000	8

The TREC robust 2004 dataset is a combined collection consisting of documents from the Financial Times, the Federal Register 94, the LA Times, and FBIS (i.e., TREC disks 4&5 with the exclusion of the Congressional Record). As in [18], we also exclude the Federal Register 94 collection because it contains few valid queries and relevant documents. We only return the top 100 documents for efficiency (the default top-k setting should be top-1000). The parameters are shown in Table II ($\gamma = 0.7$ and $\delta = 1000$).

TABLE II: Default Parameters for the TREC robust 2004 dataset

Parameter	n	m	k	α	β	ζ	κ	λ
Value	468370	400	100	1.0	1.0	0.005	45000	4

The main difference between these two parameter settings is that we set α for the TREC robust 2004 dataset to be relatively small. The reason is that for this dataset the queried keywords are regarded as more important. Most of the related documents contain the queried keywords, and we accordingly set α to a small value.

B. Semantic Accuracy Evaluation

In this section, we evaluate the semantic accuracy of different schemes from the perspective of semantic precision and recall. To define a correct search result, we use words from a specified category as the queried keywords, and the documents from the related categories constitute the correct search result. For example, terms such as "HDD" and "harddrive" are computer-related, while "playoff" and "ESPN" are sports-related. We select words with different term frequencies to simulate a user's real-world search requests. Some of the words appear in many nonrelated categories, while others only appear in the semantically related categories. We use the precision metric [15] to evaluate the search accuracy; it is calculated as follows:

$$P = \frac{TP}{TP + FP} \times 100\% \tag{36}$$

where TP and FP represent the counts of documents in the search result that, respectively, belong and do not belong to the category of the search intent.

To evaluate the recall of various schemes, we presume that TFIDF-MRSE returns all documents containing the queried keywords in the top-k search. We evaluate the number of documents returned in other schemes and regard it as the recall. A good scheme should achieve high values of both recall and precision. However, not all the documents returned by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3047921, IEEE Transactions on Cloud Computing

TFIDF-MRSE are semantically related to the search request. The recall calculation method is as follows:

$$recall = \frac{|R \cap R_{TFIDF}|}{|R_{TFIDF}|} \times 100\% \quad (37)$$

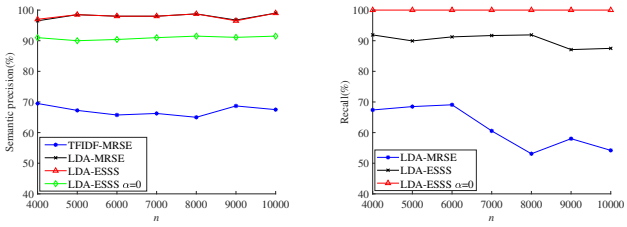
where R is the search result, and R_{TFIDF} is the search result of TFIDF-MRSE.

We also perform the experiment on LDA-ESSS when $\alpha = 0$, which makes LDA-ESSS equivalent to QL-MRSE, to test the effectiveness of the feature keywords and query likelihood model.

1) The impact of n. Fig. 4a shows that topic model based schemes outperform TFIDF-MRSE. On average, the semantic precisions of LDA-MRSE and LDA-ESSS are 97.91% and 97.96%. Because the TF-IDF model is not a semantic model, documents that are unrelated to the search intention are added to the search result due to the presence of the queried keywords. Additionally, if there are fewer than k documents containing the queried keywords, the precision is even worse. The LDA topic model utilizes topic information for searching relevant documents, and thus the presence of keywords will not affect the search result if such keywords are not topically related to the search request. LDA-ESSS has the same semantic precision as LDA-MRSE, and the feature keywords information is not reducing the search accuracy because we extract such keywords also from the perspective of topics. Moreover, the result shows that QL-MRSE (LDA-ESSS with α=0) still outperforms TFIDF-MRSE in semantic precision and achieves above 90% precision, as does LDA-ESSS, which shows that the query likelihood model has a better relevance measurement than the TF-IDF model from the perspective of semantics. However, it slightly underestimates the semantic relevance among the text.

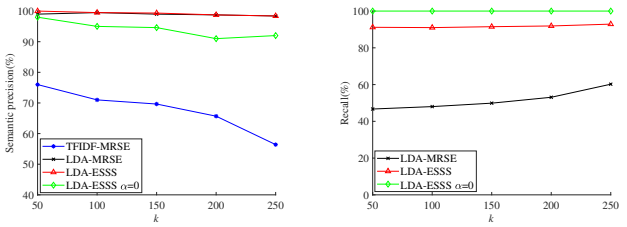
Fig. 4b indicates that LDA-ESSS greatly improves the recall of the topic model based schemes by an average of 90.18%, while LDA-MRSE only achieves 61.54%. The use of feature keywords can increase the relevance score between the query and the documents that match the queried keywords. The feature keywords extracted by LDA-ESSS have strong semantic meanings. Thus, this scheme can significantly increase recall without a loss of precision. When α is set to 0, LDA-ESSS can achieve an almost 100% recall because we select the keywords that are strongly semantically defined, and the precision could decline if the queried keywords are not all related to the search intention. However, the result shows that the keyword extraction is effective, and the user can simply set α to a moderate value to obtain more accurate results.

2) The impact of k. Fig. 5a indicates that the semantic precision of topic model based schemes is greatly higher than that of TFIDF-MRSE. Specifically, the average precisions of LDA-MRSE and LDA-ESSS are 98.93% and 99.2%. LDA-ESSS slightly outperforms LDA-MRSE, because the feature keywords can help the scheme find more suitable documents as the search result. The reason that LDA-based schemes outperform TFIDF-MRSE is that using topic information can lead to selecting the documents that better fit the user's search intention. As k increases, the precision of TFIDF-MRSE decreases significantly. This occurs because the documents



(a) Semantic precision (b) Recall

Fig. 4: Impact of n.



(a) Semantic precision (b) Recall

Fig. 5: Impact of k.

with relatively low relevance scores to the query intention are added to the search result, and such documents are mostly unrelated to the search intention.

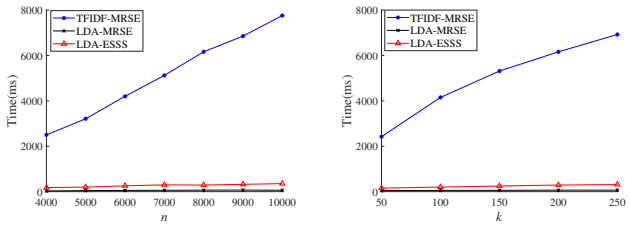
Fig. 5b indicates that the recall of LDA-ESSS is more than 90% and is better than that of LDA-MRSE that attains, on average, 50%. As k grows, the recall of LDA-MRSE increases accordingly because the documents that contain the queried keywords that have slightly lower topic relevance scores are added to the result. However, LDA-ESSS takes the presence of keywords into consideration, and it is reasonable that the documents that both have high *TRScore* and *QLScore* are ranked at the top of the search result.

C. Search Efficiency Evaluation

In this section, we evaluate the time cost of the search process in TFIDF-MRSE, LDA-MRSE and LDA-ESSS.

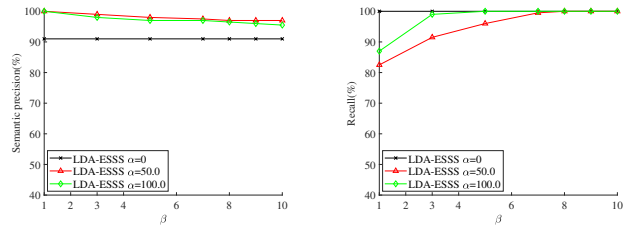
1) The impact of n. Fig. 6a indicates that the search time cost of topic model based schemes is significantly less than that of TFIDF-MRSE. On average, the time cost of LDA-MRSE and LDA-ESSS is 99% and 94.7% less than that of TFIDF-MRSE. As n increases, the time cost of three schemes increase, but the growing tendency of TFIDF-MRSE is more notably pronounced than that of LDA-based schemes. The reason is that computing the secure inner product between the document vector and the query vector consumes a large amount of time. The vector dimension in TFIDF-MRSE is the entire scale of the dictionary which is much larger than the number of topics. Additionally, LDA-ESSS only selects a few keywords from the dictionary as the feature keywords because most of the words do not have much information for representing the documents or the dataset according to the information gain. Thus, the time cost of LDA-MRSE and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3047921, IEEE Transactions on Cloud Computing



(a) Impact of n (b) Impact of k

Fig. 6: Search Efficiency Evaluation.



(a) Semantic precision (b) Recall

Fig. 7: Parameter Settings.

LDA-ESSS are better than that of TFIDF-MRSE and increase remarkably slower than that of TFIDF-MRSE.

2) Fig. 6b also shows that the search time costs of LDA-MRSE and LDA-ESSS are clearly better than that of TFIDF-MRSE. The reason is the same as 1). Additionally, the time costs of three schemes increases as k increases. The reason is that all three schemes use an index tree to improve the search efficiency, and when k increases, the search process needs to traverse more tree nodes to obtain the search result. Doing so leads to more inner product operations that increase the search time cost. However, in LDA-MRSE and LDA-ESSS, the heights of index trees are smaller because we store the document vectors in every nodes in the tree, while TFIDF-MRSE stores them only in leaf nodes. The search algorithm will access the documents earlier in LDA-MRSE and LDA-ESSS which increases the pruning effect in the greedy depth-first search algorithm.

D. Keyword Accuracy Evaluation

In this section, we evaluate the performance of our scheme on the TREC robust 2004 dataset. This dataset has predefined queries(topics): queries 301-450, queries 601-650, and queries 651-700. These queries have the corresponding relevant documents that are considered to be the correct search results. We take the topic part of these queries as our search requests. We compute the interpolated average precision and mean average precision (MAP) using $trec\_eval$ [1]. as our accuracy evaluation. Because TF-IDF performs poorly on the TREC dataset, we implement BM25-MRSE according to (1) for a more informative comparison.

[1]https://github.com/usnistgov/trec_eval

The accuracy of search results on the TREC dataset is presented in Table III with comparisons to the results of TFIDF-MRSE and BM25-MRSE.

TABLE III: Search result accuracy comparison of different ranked search schemes. The evaluation measures are interpolated average precision and MAP.

	TFIDF-MRSE	BM25-MRSE	LDA-ESSS	+ BM25
Rel	17195	17195	17195	
Rel_ret	1772	3381	**4245**	25.55%
0	0.3227	0.5714	**0.6790**	18.83%
0.1	0.1651	0.3701	**0.4681**	26.48%
0.2	0.0973	0.2525	**0.3438**	36.16%
0.3	0.0671	0.1774	**0.2560**	44.30%
0.4	0.0473	0.1246	**0.1731**	38.92%
0.5	0.0368	0.0932	**0.1315**	41.09%
0.6	0.0268	0.0671	**0.0818**	17.97%
0.7	0.0171	0.0418	**0.0667**	59.56%
0.8	0.0094	0.0199	**0.0319**	60.30%
0.9	0.0080	0.0107	**0.0129**	20.56%
1	0.0080	**0.0092**	0.0083	-9.78%
MAP	0.0574	0.1362	**0.1803**	32.38%

The result shows that our scheme outperforms both TFIDF-MRSE and BM25-MRSE. The TF-IDF model and BM25 model only consider the occurrence of queried keywords. When keywords do not appear in the given document, their relevance scores are equal to zero. However, the LDA topic model views the queried keywords from the perspective of topics and infers the search intention by the context of all the queried keywords. This results in a significant improvement from the traditional TF-IDF model.

We consider a specific example in the TREC dataset to further illustrate why LDA-ESSS outperforms keyword occurrence-based models. Query 301 is the first query in the dataset, and its queried keywords are “International Organized Crime”, i.e., the intent is to obtain the documents that discuss organizations that participate in international criminal activities. 1) The top (1^{st}) document in our LDA-ESSS scheme is “FBIS3-24143”; this document is “An interview with the head of the Regional Directorate for Fighting Organized Crime”. It is not relevant to query 301 according to the predefined result, but the document is very close to the query’s topic of “Organized Crime”. The top (1^{st}) document in BM25-MRSE is “FBIS4-20472” that discusses “Selections from the Guide to China’s Science and Technology”; it is also not relevant to the query. The content of this document is indeed entirely irrelevant to the topic of criminal organizations. When we examine the document, we find that words “crime”, “organized” and “international” all appear in the document many times but do so in a different context. Moreover, this document is far longer than average. These two reasons distort the relevance measurement of the BM25 and TF-IDF models, leading to worse rank results. 2) We analyze the first query’s relevant document in both schemes. Document “FBIS3-21961”, ranked 3^{rd} in LDA-ESSS, discusses “Views of Growth of Organized Crime in a Europe Conference”. This document is of medium length and is very relevant to the query topic. Document “FBIS3-37947”, ranked 8^{th} in BM25-MRSE, discusses the “Russian National Security Concept”. It is also rather long,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3047921, IEEE Transactions on Cloud Computing

and only a part of the document talks about organized crime.

In conclusion, without a topic representation, the keyword occurrence based models cannot truly "understand" the theme (topic) of a document. Furthermore, these models are more likely to be affected by the length of the document and the number of occurrences of keywords. LDA-ESSS considers both the keyword occurrence and topic relevance, which makes our model more accurate for the TREC dataset. Most of the queried keywords match with the extracted feature keyword dictionary, which also shows that our feature keyword extraction can lead to high usability.

E. Parameter Settings

In this section, we discuss how the parameters affect the performance of LDA-ESSS and how to reasonably set the parameters' values.

1) *α and β*

Parameters α and β can affect the search accuracy of LDA-ESSS significantly because they directly change the relevance scores between documents and the query.

Using the 20 newsgroup dataset, we perform three different experiments on the same search request when α is set to 0, 50.0 and 100.0. As Fig. 7a shows, the setting of α and β almost do not affect the semantic precision of LDA-ESSS. The reason is that we use topic information to extract the feature keywords and measure the relevance score with *QLScore* (the feature keyword relevance score). Parameter β adjusts the weight of *QLScore* that also measures the semantic relation between the query and documents. Thus, β can adjust the ranking of search results without significantly affecting the semantic relevance. When α=0, precision declines to approximately 91%; this occurs because some documents that are unrelated to the search intention have similar values of *QLScore*, and we should use topic relevance to filter them by setting α to a nonzero value.

Fig. 7b indicates that the recall of LDA-ESSS increases as β increases and tends to be stable. When α=0, the recall is almost reaching 100%. The reason is that the topic relevance score can filter the documents that contain the queried keywords but are not particularly related to the search intention, in other words, the documents have low *QLScore* and *TRScore*. When α=0, LDA-ESSS can achieve both high recall and high precision because we add topic relevance to the *QLScore* relevance measurement. Thus, even if only the query likelihood model is used, LDA-ESSS can perform well on a semantic-aware search.

For the TREC dataset, we set α to a relatively small value because we want the query likelihood model to perform the majority of relevance measurement. The effects of α and β on MAP and precision at 20 (p_20) are shown in Table IV.

TABLE IV: Effects of α and β on MAP and p_20.

$\alpha:\beta$	1:5	1:1	10:1	50:1
MAP	0.1779	**0.1803**	0.1784	0.1383
p_20	0.3221	**0.3273**	0.3225	0.2504

The result shows that when α:β is 1:1, LDA-ESSS exhibits the best performance. Additionally, when α is in an appropriate range, it only changes MAP by 1% because *TRScore* contributes much less to the final relevance than does *QLScore*. However, when α = 50, the accuracy declines significantly. The reason is that the TREC task uses short queries to perform searches, and the queried keywords are much more important than the abstract semantics. Because we have already added topic relevance in *QLScore*, we only need the query's topic information as auxiliary measurement. Thus, α is better set to a small value.

Parameters α and β balance the importance of a query's topic relevance and that of the query's keyword relevance. The data owner can easily specify these two parameters by performing a test search. In particular, the data owner can perform a search with a feature keyword from FW with a medium $P_{ML}(fw|D)$ value (it is the relevance when the keyword does not show in the given document). Having received the ranked search results, the data owner can take the most related document and obtain its *TRScore* and *QLScore*. For example, suppose that *TRScore*=0.1 and *QLScore*=-5; if the data owner sets β = 1.0 and α is set to 50, the data owner will obtain the ranked results with both high topic relevance and keyword relevance. If α is set to a small value such as 1.0, the data owner will obtain results that are more related to the queried keywords. In normal cases, we suggest that the data owner set both α and β to 1.0 because most searches are biased towards keywords, and the query likelihood model considers both keyword occurrence and semantics.

2) $|FW|$ and *Other Parameters*

The scale of feature dictionary $|FW|$ also affects the search precision of LDA-ESSS if FW does not include the queried keywords. We experimentally assess the effect of this scale on MAP and precision at 20 (p_20) for the TREC dataset, as shown in TABLE V. The result indicates that if the scale of FW is set to $\frac{1}{20}$ and $\frac{1}{10}$ of the dataset dictionary, our scheme can achieve approximately 85% and 95% search precision in regard to using the whole dictionary as FW, i.e., the scale of FW does not markedly affect the search precision. The user can use the entire dictionary as the feature keyword dictionary for a higher search precision, but search efficiency will decline significantly. Among the parameters, ζ, κ and λ are those that can change $|FW|$. Parameter ζ should be set to the most common lowest value of topic relevance among all the keywords because ζ is used to measure whether a keyword is related to a given topic. Parameter κ should be set to at least $\frac{1}{20}$ of the entire dictionary's size. Parameter λ should be set according to the average length of documents in the dataset. However, using larger κ and λ will guarantee higher recall and precision, and the data owner can adjust their values to balance accuracy and efficiency. The experiments show that γ and δ do not substantially affect the search accuracy.

TABLE V: Effect of $|FW|$ on MAP and p_20.

$\lvert FW\rvert$	$\frac{\lvert W\rvert}{20}$	$\frac{\lvert W\rvert}{10}$	$\frac{\lvert W\rvert}{4}$	$\lvert W\rvert$
MAP	0.1705	0.1803	0.1915	**0.1960**
p_20	0.2750	0.3273	0.3396	**0.3430**

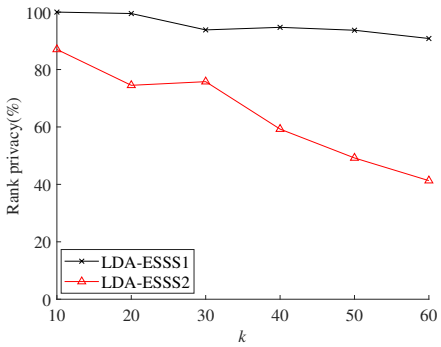


Fig. 8: Rank privacy.

F. Rank Privacy

In this section, we evaluate the rank privacy of our scheme under various parameter settings. As we discussed in Section VII.D, a curious cloud server can use the rank result lists to infer the connection between different search requests and even link them (i.e., linking trapdoors). In LDA-ESSS, the cloud server cannot use the relevance scores to link the trapdoors because different α and β will change them completely. Thus, we measure the rank privacy similarly to [8]; it is defined as

$$P_r = \sum \frac{|rank_i - rank_i'|}{k^2} \tag{38}$$

where $rank_i$ is the rank number of a document in a given scheme's top-k ranked result list, and $rank_i'$ is the rank number of the same document in the scheme with default parameters.

We assume that the cloud server can infer linkability between trapdoors based on similarity between the rank results' lists. We use LDA-ESSS with default parameters $\alpha = 50.0$ and $\beta = 1.0$ to obtain the reference rank result lists; these parameters mean that the user wants to obtain the documents that have both high topic relevance and occurrence of keywords (*QLScore*). We perform an experiment with two different parameter settings, denoted by LDA-ESSS1 and LDA-ESSS2. For LDA-ESSS1, α=1.0 and β=1.0, i.e., the user wants to obtain the documents that are highly related to the queried keywords. For LDA-ESSS2, α=200.0 and β=1.0, i.e., the user wants to obtain the documents that are highly semantically related to the queried keywords. Fig. 8 indicates that LDA-ESSS1 and LDA-ESSS2 can both achieve high rank privacy with average values of 95.41% and 64.55%. Our scheme achieves significantly higher rank privacy than does [8], which reaches approximately 22% rank privacy on average. The reason that LDA-ESSS2 has relatively low rank privacy is that the topic relevance scores *TRScore* in both LDA-ESSS and LDA-ESSS2 are the main relevance measurement in scoring the document and the query, and their rank result lists are more similar. The result shows that different α and β can enhance the security of our scheme.

IX. CONCLUSION AND FUTURE WORK

In this paper, we propose an enhanced semantic-aware multi-keyword ranked search scheme over encrypted cloud data based on the LDA topic model and the query likelihood model. By taking the keywords information into consideration, we increase the search accuracy of the LDA-based scheme. We use the LDA-based information gain and the TF-ITF model to extract the feature keywords from the document set. The original document vector is extended by the feature keyword vector that stores the query likelihood model's relevance score between the keyword and the document. In the search process, the enhanced scheme can obtain a more search request-related search result. The user can also adjust the weight parameters to obtain a more satisfying result. The presence of feature keywords can help our scheme to meet the user's search intention more accurately. We also adopt an index tree to further improve the efficiency of our scheme. The secure kNN algorithm is used to protect our scheme under two different threat models.

In this work, we first incorporate the query likelihood model into searchable encryption and propose an enhanced semantic-aware multi-keyword ranked search scheme over encrypted cloud data based on the LDA topic model. The query likelihood model is a powerful model in document retrieval; as we discuss in the security analysis, a search scheme based on such a model is more secure than the TF-IDF model-based scheme. The proposed LDA-ESSS also achieves higher search accuracy than that of the traditional TF-IDF model. We hope that our model can provide a new way to generate the document index and perform searches. The TREC robust 2004 dataset first used in our work is also suitable for measuring performance of a multi-keyword ranked search scheme. In future work, we can use the query likelihood model and the LDA topic model to implement solutions for various multi-keyword ranked search tasks, such as fuzzy keyword search, verifiable keyword search, etc. LDA-ESSS is a new model in searchable encryption, and thus needs to be explored further to demonstrates its effectiveness and usability.

X. ACKNOWLEDGEMENTS

This work is supported by NSFC 61872197, 61972209 and 61772446, PSFC 2019M651919, CCF-HUAWEI DBIR2020001A, NJUPT-NRF NY217119, and HK RGC GRF PolyU 15216220.

REFERENCES

[1] A. Swaminathan, Y. Mao, G. Su, H. Gou, A. L. Varna, S. He, M. Wu, and D. W. Oard, "Confidentiality-preserving rank-ordered search," in *Proceedings of the 2007 ACM Workshop On Storage Security And Survivability, StorageSS 2007*, Alexandria, VA, USA, Oct. 2007, pp. 7–12.

[2] D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy, S&P 2000*, Berkeley, California, USA, May 2000, pp. 44–55.

[3] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proceedings of the 2010 International Conference on Distributed Computing Systems, ICDCS 2010*, Genova, Italy, Jun. 2010, pp. 253–262.

[4] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber^{+r}: top-k retrieval from a confidential index," in *Proceedings of the 12th International Conference on Extending Database Technology, EDBT 2009*, Saint Petersburg, Russia, Mar. 2009, pp. 439–449.

[5] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proceedings of the 30th IEEE International Conference on Computer Communications, INFOCOM 2011*, Shanghai, China, Apr. 2011, pp. 829–837.

[6] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 3025–3035, 2014.

[7] 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 Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 951–963, 2016.

[8] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[9] X. Zhu, H. Dai, X. Yi, G. Yang, and X. Li, "MUSE: an efficient and accurate verifiable privacy-preserving multikeyword text search over encrypted cloud data," *Security and Communication Networks*, vol. 2017, pp. 1–17, 2017.

[10] C. Tseng, C. Lu, and C. Chou, "Efficient privacy-preserving multi-keyword ranked search utilizing document replication and partition," in *Proceedings of the 12th Annual IEEE Consumer Communications and Networking Conference, CCNC 2015*, Las Vegas, NV, USA, Jan. 2015, pp. 671–676.

[11] S. Tahir, S. Ruj, Y. Rahulamathavan, M. Rajarajan, and C. Glackin, "A new secure and lightweight searchable encryption scheme over encrypted cloud data," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 4, pp. 530–544, 2019.

[12] X. Jiang, J. Yu, J. Yan, and R. Hao, "Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data," *Information Sciences*, vol. 403, pp. 22–41, 2017.

[13] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009*, Providence, Rhode Island, USA, Jun. 2009, pp. 139–152.

[14] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," in *Proceedings of the Third Text REtrieval Conference, TREC 1994*, Gaithersburg, Maryland, USA, Nov. 1994, pp. 109–126.

[15] M. Gabryel, R. Damasevicius, and K. Przybyszewski, "Application of the bag-of-words algorithm in classification the quality of sales leads," in *Proceedings of Artificial Intelligence and Soft Computing - 17th International Conference, ICAISC 2018*, Zakopane, Poland, Jun. 2018, pp. 615–622.

[16] V. Lavrenko and W. B. Croft, "Relevance-based language models," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2001*, New Orleans, Louisiana, USA, Sep. 2001, pp. 120–127.

[17] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,SIGIR '98*, Melbourne, Australia, Aug. 1998, pp. 275–281.

[18] X. Wei and W. B. Croft, "LDA-based document models for ad-hoc retrieval," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2006*, Seattle, Washington, USA, Aug. 2006, pp. 178–185.

[19] S. MacAvaney, F. M. Nardini, R. Perego, N. Tonellotto, N. Goharian, and O. Frieder, "Efficient document re-ranking for transformers by precomputing term representations," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*, Virtual Event, China, 2020, pp. 49–58.

[20] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, Minneapolis, MN, USA, 2019, pp. 4171–4186.

[21] P. Scheuermann and A. M. Ouksel, "Multidimensional B-trees for associative searching in database systems," *Information Systems*, vol. 7, no. 2, pp. 123–137, 1982.

[22] Z. Xia, Y. Zhu, X. Sun, and L. Chen, "Secure semantic expansion based search over encrypted cloud data supporting similarity ranking," *Journal of Cloud Computing*, vol. 3, p. 8, 2014.

[23] 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 Transactions on Consumer Electronics*, vol. 60, no. 1, pp. 164–172, 2014.

[24] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1874–1884, 2017.

[25] Z. Fu, L. Xia, X. Sun, A. X. Liu, and G. Xie, "Semantic-aware searching over encrypted data for cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2359–2371, 2018.

[26] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[27] G. Heinrich, "Parameter estimation for text analysis," vsonix GmbH and University of Leipzig, Technical Report Version 2.9, 2009.

[28] H. Pang, X. Xiao, and J. Shen, "Obfuscating the topical intention in enterprise text search," in *Proceedings of the IEEE 28th International Conference on Data Engineering, ICDE 2012*, Washington, DC, USA, Apr. 2012, pp. 1168–1179.

[29] P. Wang and C. V. Ravishankar, "On masking topical intent in keyword search," in *Proceedings of the IEEE 30th International Conference on Data Engineering, ICDE 2014*, Chicago, IL, USA, Mar. 2014, pp. 256–267.

[30] H. Dai, X. Dai, X. Yi, G. Yang, and H. Huang, "Semantic-aware multi-keyword ranked search scheme over encrypted cloud data," *Journal of Network and Computer Applications*, vol. 147, pp. 1–11, 2019.

[31] C. Davis, "The norm of the Schur product operation," *Numerische Mathematik*, vol. 4, no. 1, pp. 343–344, Dec. 1962.

[32] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.

[33] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13*, Hangzhou, China, May 2013, pp. 71–82.

[34] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1467–1479, 2012.

[35] H. Delfs and H. Knebl, *Introduction to Cryptography - Principles and Applications, Third Edition*. Berlin, Germany: Springer, 2015.

[36] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[37] M. Bulut, "ReducedCBT and SuperCBT, Two New and Improved Complete Binary Tree Structures," *CoRR*, vol. abs/1401.7741, 2014.

[38] K. Lang, "Newsweeder: Learning to Filter Netnews," in *Proceedings of the Twelfth International Conference on Machine Learning, ICML 1995*, Tahoe City, California, USA, Jul. 1995, pp. 331–339.

[39] E. M. Voorhees and L. P. Buckland, Eds., *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004*, vol. 500-261. National Institute of Standards and Technology (NIST).