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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

Enabling Fine-grained Multi-keyword Search Supporting Classified Sub-dictionaries over Encrypted Cloud Data

Hongwei Li, *Member, IEEE,* Yi Yang, *Student Member, IEEE,* Tom H. Luan, *Member, IEEE,* Xiaohui Liang, *Student Member, IEEE,* Liang Zhou, *Member, IEEE,* and Xuemin (Sherman) Shen, *Fellow, IEEE*

Abstract—Using cloud computing, individuals can store their data on remote servers and allow data access to public users through the cloud servers. As the outsourced data are likely to contain sensitive privacy information, they are typically encrypted before uploaded to the cloud. This, however, significantly limits the usability of outsourced data due to the difficulty of searching over the encrypted data. In this paper, we address this issue by developing the fine-grained multi-keyword search schemes over encrypted cloud data. Our original contributions are three-fold. First, we introduce the relevance scores and preference factors upon keywords which enable the precise keyword search and personalized user experience. Second, we develop a practical and very efficient multi-keyword search scheme. The proposed scheme can support complicated logic search the mixed "AND", "OR" and "NO" operations of keywords. Third, we further employ the classified sub-dictionaries technique to achieve better efficiency on index building, trapdoor generating and query. Lastly, we analyze the security of the proposed schemes in terms of confidentiality of documents, privacy protection of index and trapdoor, and unlinkability of trapdoor. Through extensive experiments using the real-world dataset, we validate the performance of the proposed schemes. Both the security analysis and experimental results demonstrate that the proposed schemes can achieve the same security level comparing to the existing ones and better performance in terms of functionality, query complexity and efficiency.

Index Terms—Searchable encryption, Multi-keyword, Fine-grained, Cloud computing.

1 INTRODUCTION

THE cloud computing treats computing as a utility and leases out the computing and storage capacities to the public individuals [1], [2], [3]. In such a framework, the individual can remotely store her data on the cloud server, namely data outsourcing, and then make the cloud data open for public access through the cloud server. This represents a more scalable, low-cost and stable way for public data access because of the scalability and high efficiency of cloud servers, and therefore is favorable to small enterprises.

- *H. Li and Y. Yang are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China (e-mail: hongweili@uestc.edu.cn; yangyi.buku@gmail.com).*
- *H. Li is with State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093) (e-mail: hongweili@uestc.edu.cn).*
- *T. Luan is with the School of Information Technology, Deakin University, Melbourne, Australia(e-mail: tom.luan@deakin.edu.au).*
- *X. Liang is with the Department of Computer Science, Dartmouth College, Hanover, USA(e-mail: Xiaohui.Liang@dartmouth.edu).*
- *L. Zhou is with the National Key Laboratory of Science and Technology on Communication, University of Electronic Science and Technology of China, China(e-mail: lzhou@uestc.edu.cn).*
- *X. Shen is with the Department of Electrical and Computer Engineering, University of Waterloo,Waterloo, Ontario, Canada(e-mail: sshen@uwaterloo.ca).*

Note that the outsourced data may contain sensitive privacy information. It is often necessary to encrypt the private data before transmitting the data to the cloud servers [4], [5]. The data encryption, however, would significantly lower the usability of data due to the difficulty of searching over the encrypted data [6]. Simply encrypting the data may still cause other security concerns. For instance, Google Search uses SSL (Secure Sockets Layer) to encrypt the connection between search user and Google server when private data, such as documents and emails, appear in the search results [7]. However, if the search user clicks into another website from the search results page, that website may be able to identify the search terms that the user has used.

On addressing above issues, the searchable encryption (e.g., [8], [9], [10]) has been recently developed as a fundamental approach to enable searching over encrypted cloud data, which proceeds the following operations. Firstly, the data owner needs to generate several keywords according to the outsourced data. These keywords are then encrypted and stored at the cloud server. When a search user needs to access the outsourced data, it can select some relevant keywords and send the ciphertext of the selected keywords to the cloud server. The cloud server then uses the ciphertext to match the outsourced encrypted keywords, and lastly returns the matching results to the search user. To achieve the similar search efficiency and precision over encrypted data as that of plaintext keyword search, an extensive body of research has been developed in literature. Wang et al. [11] propose a ranked keyword search scheme which considers the relevance scores

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

of keywords. Unfortunately, due to using order-preserving encryption (OPE) [12] to achieve the ranking property, the proposed scheme cannot achieve unlinkability of trapdoor. Later, Sun et al. [13] propose a multi-keyword text search scheme which considers the relevance scores of keywords and utilizes a multidimensional tree technique to achieve efficient search query. Yu et al. [14] propose a multi-keyword top-k retrieval scheme which uses fully homomorphic encryption to encrypt the index/trapdoor and guarantees high security. Cao et al. [6] propose a multi-keyword ranked search (MRSE), which applies coordinate machine as the keyword matching rule, i.e., return data with the most matching keywords.

Although many search functionalities have been developed in previous literature towards precise and efficient searchable encryption, it is still difficult for searchable encryption to achieve the same user experience as that of the plaintext search, like Google search. This mainly attributes to following two issues. Firstly, query with user preferences is very popular in the plaintext search [15], [16]. It enables personalized search and can more accurately represent user's requirements, but has not been thoroughly studied and supported in the encrypted data domain. Secondly, to further improve the user's experience on searching, an important and fundamental function is to enable the multi-keyword search with the comprehensive logic operations, i.e., the "AND", "OR" and "NO" operations of keywords. This is fundamental for search users to prune the searching space and quickly identify the desired data. Cao et al. [6] propose the coordinate matching search scheme (MRSE) which can be regarded as a searchable encryption scheme with "OR" operation. Zhang et al. [17] propose a conjunctive keyword search scheme which can be regarded as a searchable encryption scheme with "AND" operation with the returned documents matching all keywords. However, most existing proposals can only enable search with single logic operation, rather than the mixture of multiple logic operations on keywords, which motivates our work.

In this work, we address above two issues by developing two Fine-grained Multi-keyword Search (FMS) schemes over encrypted cloud data. Our original contributions can be summarized in three aspects as follows:

- We introduce the relevance scores and the preference factors of keywords for searchable encryption. The relevance scores of keywords can enable more precise returned results, and the preference factors of keywords represent the importance of keywords in the search keyword set specified by search users and correspondingly enables personalized search to cater to specific user preferences. It thus further improves the search functionalities and user experience.
- We realize the "AND", "OR" and "NO" operations in the multi-keyword search for searchable encryption. Compared with schemes in [6], [13] and [14], the proposed scheme can achieve more comprehensive functionality and lower query complexity.
- We employ the classified sub-dictionaries technique to enhance the efficiency of the above two schemes. Extensive experiments demonstrate that the enhanced schemes can achieve better efficiency in terms of index building, trapdoor generating and query in the comparison with schemes in [6], [13] and [14].

The remainder of this paper is organized as follows. In Section 2, we outline the system model, threat model, security requirements and design goals. In Section 3, we describe the preliminaries of the proposed schemes. We present the developed schemes and enhanced schemes in details in Section 4 and Section 5, respectively. Then we carry out the security analysis and performance evaluation in Section 6 and Section 7, respectively. Section 8 provides a review of the related works and Section 9 concludes the paper.

2 SYSTEM MODEL, THREAT MODEL AND SECURITY REQUIREMENTS

2.1 System Model

As shown in Fig. 1, we consider a system consists of three entities.

- *Data owner*: The data owner outsources her data to the cloud for convenient and reliable data access to the corresponding search users. To protect the data privacy, the data owner encrypts the original data through symmetric encryption. To improve the search efficiency, the data owner generates some keywords for each outsourced document. The corresponding index is then created according to the keywords and a secret key. After that, the data owner sends the encrypted documents and the corresponding indexes to the cloud, and sends the symmetric key and secret key to search users.
- *Cloud server*: The cloud server is an intermediate entity which stores the encrypted documents and corresponding indexes that are received from the data owner, and provides data access and search services to search users. When a search user sends a keyword trapdoor to the cloud server, it would return a collection of matching documents based on certain operations.
- *Search user*: A search user queries the outsourced documents from the cloud server with following three steps. First, the search user receives both the secret key and symmetric key from the data owner. Second, according to the search keywords, the search user uses the secret key to generate trapdoor and sends it to the cloud server. Last, she receives the matching document collection from the cloud server and decrypts them with the symmetric key.

2.2 Threat Model and Security Requirements

In our threat model, the cloud server is assumed to be "honest-but-curious", which is the same as most related works on secure cloud data search [13], [14], [6]. Specifically, the cloud server honestly follows the designated protocol specification. However, the cloud server could be "curious" to infer and analyze data (including index) in its storage and message flows received during the protocol so as to learn additional information. we consider two threat models depending on the information available to the cloud server, which are also used in [13], [6].

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

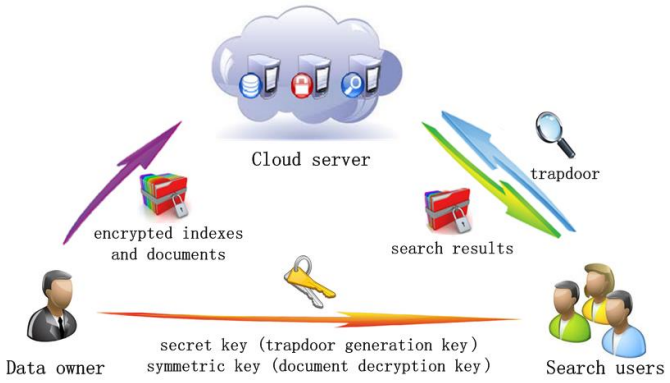

Fig. 1. System model

- **Known Ciphertext Model:** The cloud server can only know encrypted document collection $\mathcal{C}$ and index collection $\mathcal{I}$, which are both outsourced from the data owner.
- **Known Background Model:** The cloud server can possess more knowledge than what can be accessed in the known ciphertext model, such as the correlation relationship of trapdoors and the related statistical of other information, i.e., the cloud server can possess the statistical information from a known comparable dataset which bears the similar nature to the targeting dataset.

Similar to [13], [6], we assume search users are trusted entities, and they share the same symmetric key and secret key. Search users have pre-existing mutual trust with the data owner. For ease of illustration, we do not consider the secure distribution of the symmetric key and the secret key between the data owner and search users; it can be achieved through regular authentication and secure channel establishment protocols based on the prior security context shared between search users and the data owner [18]. In addition, to make our presentations more focused, we do not consider following issues, including the access control problem on managing decryption capabilities given to users and the data collection's updating problem on inserting new documents, updating existing documents, and deleting existing documents, are separated issues. The interested readers on above issues may refer to [6], [5], [10], [19].

Based on the above threat model, we define the security requirements as follows:

- *Confidentiality of documents*: The outsourced documents provided by the data owner are stored in the cloud server. If they match the search keywords, they are sent to the search user. Due to the privacy of documents, they should not be identifiable except by the data owner and the authorized search users.
- *Privacy protection of index and trapdoor*: As discussed in Section 2.1, the index and the trapdoor are created based on the documents' keywords and the search keywords, respectively. If the cloud server identifies the content of index or trapdoor, and further deduces any association between keywords and encrypted documents, it may learn the major subject of a document, even the content of a short document [20]. Therefore, the content of index and trapdoor cannot be identified by the cloud server.
- *Unlinkability of trapdoor*: The documents stored in the cloud server may be searched many times. The cloud server should not be able to learn any keyword information according to the trapdoors, e.g., to determine two trapdoors which are originated from the same keywords. Otherwise, the cloud server can deduce relationship of trapdoors, and threaten to the privacy of keywords. Hence the trapdoor generation function should be randomized, rather than deterministic. Even in case that two search keyword sets are the same, the trapdoors should be different.

3 PRELIMINARIES

In this section, we define the notation and review the secure kNN computation and relevance score, which will serve as the basis of the proposed schemes.

3.1 Notation

- $\mathcal{F}$—the document collection to be outsourced, denoted as a set of N documents $\mathcal{F} = (F_1, F_2, \cdots, F_N)$.
- $\mathcal{C}$—the encrypted document collection according to $\mathcal{F}$, denoted as a set of N documents $\mathcal{C} = (C_1, C_2, \cdots, C_N)$.
- $\mathcal{FID}$—the identity collection of encrypted documents $\mathcal{C}$, denoted as $\mathcal{FID} = (FID_1, FID_2, \cdots, FID_N)$.
- $\mathcal{W}$—the keyword dictionary, including m keywords, denoted as $\mathcal{W} = (w_1, w_2, \cdots, w_m)$.
- $\mathcal{I}$—the index stored in the cloud server, which is built from the keywords of each document, denoted as $\mathcal{I} = (I_1, I_2, \cdots, I_N)$.
- $\widetilde{\mathcal{W}}$—the query keyword set generated by a search user, which is a subset of $\mathcal{W}$.
- $T_{\widetilde{\mathcal{W}}}$—the trapdoor for keyword set $\widetilde{\mathcal{W}}$.
- $\widetilde{\mathcal{FID}}$—the identity collection of documents returned to the search user.
- FMS(CS)—the abbreviation of FMS and FMSCS.

3.2 Secure kNN Computation

We adopt the work of Wong et al. [21] as our foundation. Wong et al. propose a secure k-nearest neighbor (kNN) scheme which can confidentially encrypt two vectors and compute Euclidean distance of them. Firstly, the secret key (S, M_1, M_2) should be generated. The binary vector S is a splitting indicator to split plaintext vector into two random vectors, which can confuse the value of plaintext vector. And M_1 and M_2 are used to encrypt the split vectors. The correctness and security of secure kNN computation scheme can be referred to [21].

3.3 Relevance Score

The relevance score between a keyword and a document represents the frequency that the keyword appears in the document. It can be used in searchable encryption for returning ranked results. A prevalent metric for evaluating the relevance score is TF $\times$ IDF, where TF (term frequency) represents the frequency of a given keyword in a document and IDF

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

(inverse document frequency) represents the importance of keyword within the whole document collection. Without loss of generality, we select a widely used expression in [22] to evaluate the relevance score as

$$Score(\widetilde{\mathcal{W}}, F_j) = \sum_{w \in \widetilde{\mathcal{W}}} \frac{1}{|F_j|} \cdot (1 + \ln f_{j,w}) \cdot \ln(1 + \frac{N}{f_w}) \quad (1)$$

where $f_{j,w}$ denotes the TF of keyword w in document F_j; f_w denotes the number of documents contain keyword w; N denotes the number of documents in the collection; and $|F_j|$ denotes the length of F_j, obtained by counting the number of indexed keywords.

4 PROPOSED SCHEMES

In this section, we firstly propose a variant of the secure kNN computation scheme, which serves as the basic framework of our schemes. Furthermore, we describe two variants of our basic framework and the corresponding functionalities of them in detail.

4.1 Basic Framework

The secure kNN computation scheme uses Euclidean distance to select k nearest database records. In this section, we present a variant of the secure kNN computation scheme to achieve the searchable encryption property.

4.1.1 Initialization

The data owner randomly generates the secret key $K = (S, M_1, M_2)$, where S is a $(m+1)$-dimensional binary vector, M_1 and M_2 are two $(m+1) \times (m+1)$ invertible matrices, respectively, and m is the number of keywords in $\mathcal{W}$. Then the data owner sends (K, sk) to search users through a secure channel, where sk is the symmetric key used to encrypt documents outsourced to the cloud server.

4.1.2 Index building

The data owner firstly utilizes symmetric encryption algorithm (e.g., AES) to encrypt the document collection $(F_1, F_2, \cdots, F_N)$ with the symmetric key sk [23], the encrypted document collection are denoted as $C_j (j = 1, 2, \cdots, N)$. Then the data owner generates an m-dimensional binary vector P according to $C_j (j = 1, 2, \cdots, N)$, where each bit $P[i]$ indicates whether the encrypted document contains the keyword w_i, i.e., $P[i] = 1$ indicates yes and $P[i] = 0$ indicates no. Then she extends P to a $(m+1)$-dimensional vector P', where $P'[m+1] = 1$. The data owner uses vector S to split P' into two $(m+1)$-dimensional vectors (p_a, p_b), where the vector S functions as a splitting indicator. Namely, if $S[i] = 0 (i = 1, 2, \cdots, m+1)$, $p_a[i]$ and $p_b[i]$ are both set as $P'[i]$; if $S[i] = 1 (i = 1, 2, \cdots, m+1)$, the value of $P'[i]$ will be randomly split into $p_a[i]$ and $p_b[i]$ ($P'[i] = p_a[i] + p_b[i]$). Then, the index of encrypted document C_j can be calculated as $I_j = (p_a M_1, p_b M_2)$. Finally, the data owner sends $C_j || FID_j || I_j \ (j = 1, 2, \cdots, N)$ to the cloud server.

4.1.3 Trapdoor generating

The search user firstly generates the keyword set $\widetilde{\mathcal{W}}$ for searching. Then, she creates a m-dimensional binary vector Q according to $\widetilde{\mathcal{W}}$, where $Q[i]$ indicates whether the i-th keyword of dictionary w_i is in $\widetilde{\mathcal{W}}$, i.e., $Q[i] = 1$ indicates yes and $Q[i] = 0$ indicates no. Further, the search user extends Q to a $(m+1)$-dimensional vector Q', where $Q'[m+1] = -s$ (the value of $-s$ will be defined in the following schemes in detail). Next, the search user chooses a random number $r > 0$ to generate $Q'' = r \cdot Q'$. Then she splits Q'' into two $(m+1)$ vectors (q_a, q_b): if $S[i] = 0 (i = 1, 2, \cdots, m+1)$, the value of $Q''[i]$ will be randomly split into $q_a[i]$ and $q_b[i]$; if $S[i] = 1 (i = 1, 2, \cdots, m+1)$, $q_a[i]$ and $q_b[i]$ are both set as $Q''[i]$. Thus, the search trapdoor $T_{\widetilde{\mathcal{W}}}$ can be generated as $(M_1^{-1} q_a, M_2^{-1} q_b)$. Then the search user sends $T_{\widetilde{\mathcal{W}}}$ to the cloud server.

4.1.4 Query

With the index $I_j (j = 1, 2, \cdots, N)$ and trapdoor $T_{\widetilde{\mathcal{W}}}$, the cloud server calculates the query result as

$$\begin{aligned} R_j = I_j \cdot T_{\widetilde{\mathcal{W}}} &= (p_a M_1, p_b M_2) \cdot (M_1^{-1} q_a, M_2^{-1} q_b) \\ &= p_a \cdot q_a + p_b \cdot q_b = P' \cdot Q'' \\ &= r P' \cdot Q' = r \cdot (P \cdot Q - s) \end{aligned} \quad (2)$$

If $R_j > 0$, the corresponding document identity FID_j will be returned.

Discussions: The Basic Framework has defined the fundamental system structure of the developed schemes. Based on the secure kNN computation scheme [21], the complementary random parameter r further enhances the security. Different values for parameter s and vectors P and Q can lead to new variants of the Basic Framework. This will be elaborated in the follows.

4.2 FMS_I

In the Basic Framework, P is a m-dimensional binary vector, and each bit $P[i]$ indicates whether the encrypted document contains the keyword w_i. In the FMS_I, the data owner first calculates the relevance score between the keyword w_i and document F_j. The relevance score can be calculated as follows:

$$Score(w_i, F_j) = \frac{1}{|F_j|} \cdot (1 + \ln f_{j,w_i}) \cdot \ln(1 + \frac{N}{f_{w_i}}) \quad (3)$$

where f_{j,w_i} denotes the TF of keyword w_i in document F_j; f_{w_i} denotes the number of documents contain keyword w_i; N denotes the number of documents in the collection; and $|F_j|$ denotes the length of F_j, obtained by counting the number of indexed keywords.

Then the data owner replaces the value of $P[i]$ with the corresponding relevance score. On the other hand, we also consider the preference factors of keywords. The preference factors of keywords indicate the importance of keywords in the search keyword set personalized defined by the search user. For a search user, he may pay more attention to the preference factors of keywords defined by himself than the relevance scores of the keywords. Thus, our goal is that

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

if a document has a keyword with larger preference factor than other documents, it should have a higher priority in the returned $\widetilde{\mathcal{FID}}$; and for two documents, if their largest preference factor keywords are the same, the document with higher relevance score of the keyword is the better matching result.

As shown in Fig. 2, we replace the values of $P[i]$ and $Q[i]$ by the relevance score and the preference factor of a keyword, respectively (thus P and Q are no longer binary). The search user can dynamically adjust the preference factors to achieve a more flexible search. For convenience, the score is rounded up, i.e., $Score(w_i, F_j) = \lceil 10 * Score(w_i, F_j) \rceil$, and we assume the relevance score is not more than D, i.e., $Score(w_i, F_j) < D$. For the search keyword set $\widetilde{\mathcal{W}} = (w_{n_1}, w_{n_2}, \cdots, w_{n_l})(1 \leq n_1 < n_2 < \cdots < n_l \leq m)$ which is ordered by ascending importance, the search user randomly chooses a super-increasing sequence $(d_1 > 0, d_2, \cdots, d_l)$ (i.e., $\sum_{i=1}^{j-1} d_i \cdot D < d_j (j = 2, 3, \cdots, l)$), where d_i is the preference factor of keyword w_{n_i}. Then the search result would be:

$$R_j = r \cdot (P \cdot Q - s) = r \cdot (\sum_{i=1}^{l} Score(w_{n_i}, F_j) \cdot d_i - s) \quad (4)$$

Theorem 1: (Correctness) For the search keyword set $\widetilde{\mathcal{W}} = (w_{n_1}, w_{n_2}, \cdots, w_{n_l})(1 \leq n_1 < n_2 < \cdots < n_l \leq m)$ which is ordered by ascending preference factors, if F_1 contains a larger preference factor keyword compared with F_2, then F_1 has higher priority in the returned $\widetilde{\mathcal{FID}}$.

Proof: For the search keyword set $\widetilde{\mathcal{W}} = (w_{n_1}, w_{n_2}, \cdots, w_{n_l})$, assume the keyword sets F_1 and F_2 contain in $\widetilde{\mathcal{W}}$ are denoted as $\widetilde{\mathcal{W}_1} = (w_{n_i}, \cdots, w_{n_x})(n_1 \leq n_i < \cdots < n_x \leq n_l)$ and $\widetilde{\mathcal{W}_2} = (w_{n_j}, \cdots, w_{n_y})(n_1 \leq n_j < \cdots < n_y \leq n_l)$, respectively, where $\widetilde{\mathcal{W}_1}$ and $\widetilde{\mathcal{W}_2}$ are both ordered by ascending preference factors, and $n_x > n_y$. As stated above, $Score(w_{n_x}, F_j) \geq 1$ since the score is rounded up, and $\sum_{i=1}^{j-1} d_i \cdot D < d_j (j = 2, 3, \cdots, l)$. Therefore, there will be

$$\begin{aligned} R_2 &= r \cdot (\sum_{w_{n_j} \in \widetilde{\mathcal{W}_2}} Score(w_{n_j}, F_2) \cdot d_j - s) \\ &< r \cdot (\sum_{j=1}^{y} Score(w_{n_j}, F_2) \cdot d_j - s) \\ &< r \cdot (\sum_{j=1}^{y} D \cdot d_j - s) < r \cdot (d_x - s) \\ &< r \cdot (Score(w_{n_x}, F_1) \cdot d_x - s) \\ &< r \cdot (\sum_{w_{n_i} \in \widetilde{\mathcal{W}_1}} Score(w_{n_i}, F_1) \cdot d_i - s) \\ &< R_1 \end{aligned} \quad (5)$$

Therefore, F_1 has higher priority in the returned $\widetilde{\mathcal{FID}}$.

Theorem 2: (Correctness) For the search keyword set $\widetilde{\mathcal{W}} = (w_{n_1}, w_{n_2}, \cdots, w_{n_l})(1 \leq n_1 < n_2 < \cdots < n_l \leq m)$ which is ordered by ascending preference factors, if the largest preference factor keyword F_1 contains is the same as that F_2 contains, and F_1 have the higher relevance score of the keyword, then F_1 have higher priority in the returned $\widetilde{\mathcal{FID}}$.

Proof: For the search keyword set $\widetilde{\mathcal{W}} = (w_{n_1}, w_{n_2}, \cdots, w_{n_l})$, assume the keyword sets F_1 and F_2 contain are denoted as $\widetilde{\mathcal{W}_1} = (w_{n_i}, \cdots, w_{n_x})(n_1 \leq n_i < \cdots < n_x \leq n_l)$ and $\widetilde{\mathcal{W}_2} = (w_{n_j}, \cdots, w_{n_x})(n_1 \leq n_j < \cdots < n_x \leq n_l)$, respectively, where $\widetilde{\mathcal{W}_1}$ and $\widetilde{\mathcal{W}_2}$ are both ordered by ascending preference factors and $Score(w_{n_x}, F_1) - Score(w_{n_x}, F_2) \geq 1$. Thus, there will be

$$\begin{aligned} R_1 =& r \cdot (\sum_{w_{n_i} \in \widetilde{\mathcal{W}_1}} Score(w_{n_i}, F_1) \cdot d_i - s) \\ &\geq r \cdot (Score(w_{n_x}, F_1) \cdot d_x - s) \end{aligned} \quad (7)$$

$$\begin{aligned} R_2 =& r \cdot (\sum_{w_{n_j} \in \widetilde{\mathcal{W}_2}} Score(w_{n_j}, F_2) \cdot d_j - s) \\ =& r \cdot (Score(w_{n_x}, F_2) \cdot d_x \\ &+ \sum_{w_{n_j} \in \widetilde{\mathcal{W}_2} - w_{n_x}} Score(w_{n_j}, F_2) \cdot d_j - s) \\ <& r \cdot (Score(w_{n_x}, F_2) \cdot d_x + \sum_{w_{n_j} \in \widetilde{\mathcal{W}_2} - w_{n_x}} D \cdot d_j - s) \\ <& r \cdot (Score(w_{n_x}, F_2) \cdot d_x + d_x - s) \end{aligned} \quad (8)$$

$$\begin{aligned} R_1 - R_2 &> r \cdot ((Score(w_{n_x}, F_1) - Score(w_{n_x}, F_2)) \cdot d_x - d_x) \\ &> r \cdot (d_x - d_x) \\ &> 0 \end{aligned} \quad (9)$$

Therefore, F_1 have higher priority in the returned $\widetilde{\mathcal{FID}}$ than F_2.

Example. We present a concrete example to help understand *Theorem 2*. The example also illustrates the working process of FMS_I. Specifically, we assume that the search keyword set is $\widetilde{\mathcal{W}} = (w_{n_1}, w_{n_2}, \cdots, w_{n_5})$, and the largest preference factor keyword of sets F_1 and F_2 is the same, which is w_{n_4}. In addition, we assume the keyword sets F_1 and F_2 are $\widetilde{\mathcal{W}_1} = (w_{n_2}, w_{n_3}, w_{n_4})$ and $\widetilde{\mathcal{W}_2} = (w_{n_1}, w_{n_3}, w_{n_4})$ respectively. Furthermore, we assume that the relevance score is not more than $D = 5$, and specially, let $Score(w_{n_4}, F_1) = 4$ and $Score(w_{n_4}, F_2) = 2$, which satisfy $Score(w_{n_4}, F_1) - Score(w_{n_4}, F_2) = 2 \geq 1$. we randomly choose a super-increasing sequence $d_i = \{1, 10, 60, 500, 3000\}(i = 1, \cdots, 5)$, for arbitrary $r > 0$, there will be

$$\begin{aligned} R_1 =& r \cdot (\sum_{w_{n_i} \in \widetilde{\mathcal{W}_1}} Score(w_{n_i}, F_1) \cdot d_i - s) \\ \geq& r \cdot (Score(w_{n_4}, F_1) \cdot d_4 - s) \\ \geq& r \cdot (4 \cdot 500 - s) \\ \geq& r \cdot (2000 - s) \end{aligned} \quad (11)$$

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

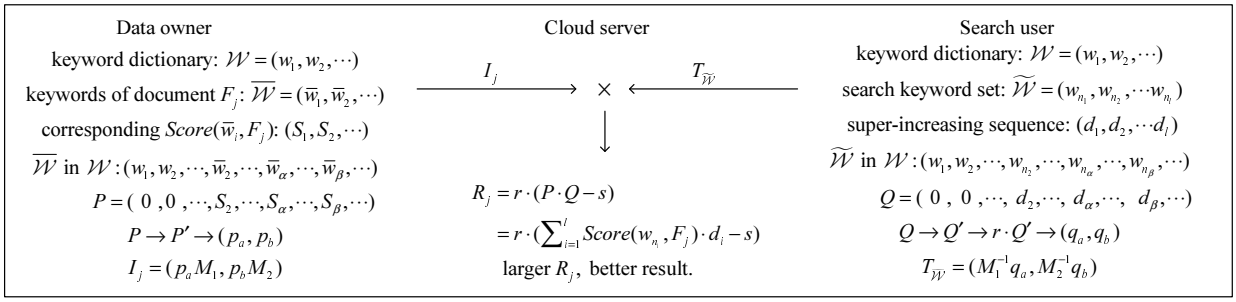

Fig. 2. Structure of the FMS_I

$$
\begin{aligned}
R_2 =& r\cdot(\sum_{w_{n_j}\in\widetilde{\mathcal{W}_2}} Score(w_{n_j},F_2)\cdot d_j - s) \qquad (12)\\
=& r\cdot(Score(w_{n_4},F_2)\cdot d_4\\
&+\sum_{w_{n_j}\in\widetilde{\mathcal{W}_2}-w_{n_4}} Score(w_{n_j},F_2)\cdot d_j - s)\\
<& r\cdot(Score(w_{n_4},F_2)\cdot d_x+\sum_{w_{n_j}\in\widetilde{\mathcal{W}_2}-w_{n_4}} D\cdot d_j - s)\\
<& r\cdot(Score(w_{n_4},F_2)\cdot d_4+d_4-s)\\
<& r\cdot(2\cdot 500+500-s)\\
<& r\cdot(1500-s)
\end{aligned}
$$

$$
\begin{aligned}
R_1-R_2 >& r\cdot(2000-s)-r\cdot(1500-s) \qquad (13)\\
>& r\cdot(2000-1500)\\
>& 500\cdot r>0
\end{aligned}
$$

4.3 FMS_II

In the FMS_II, we do not change the vector P in the Basic Framework, but replace the value of $Q[i]$ by the weight of search keywords, as shown in Fig. 3. With the weight of keywords, we can also implement some operations like "OR", "AND" and "NO" in the Google Search to the searchable encryption.

Assume that the keyword sets corresponding to the "OR", "AND" and "NO" operations are $(w'_1,w'_2,\cdots,w'_{l_1})$, $(w''_1,w''_2,\cdots,w''_{l_2})$ and $(w'''_1,w'''_2,\cdots,w'''_{l_3})$, respectively. Denote "OR", "AND" and "NO" operations by $\vee$, $\wedge$ and $\neg$, respectively. Thus the matching rule can be represented as $(w'_1\vee w'_2\vee\cdots\vee w'_{l_1})\wedge(w''_1\wedge w''_2\wedge\cdots\wedge w''_{l_2})\wedge(\neg w'''_1\wedge\neg w'''_2\wedge\cdots\wedge\neg w'''_{l_3})$. For "OR" operation, the search user chooses a super-increasing sequence $(a_1>0,a_2,\cdots,a_{l_1})(\sum_{k=1}^{j-1}a_k<a_j(j=2,\cdots,l_1))$ to achieve searching with keyword weight. To enable searchable encryption with "AND" and "NO" operations, the search user chooses a sequence $(b_1,b_2,\cdots,b_{l_2},c_1,c_2,\cdots,c_{l_3})$, where $\sum_{k=1}^{l_1}a_k<b_h(h=1,2,\cdots,l_2)$ and $\sum_{k=1}^{l_1}a_k+\sum_{h=1}^{l_2}b_h<c_i(i=1,2,\cdots,l_3)$. Assume $(w'_1,w'_2,\cdots,w'_{l_1})$ are ordered by ascending importance, then according to the search keyword set $(w'_1,w'_2,\cdots,w'_{l_1},w''_1,w''_2,\cdots,w''_{l_2},w'''_1,w'''_2,\cdots,w'''_{l_3})$, the corresponding values in Q are set as $(a_1,a_2,\cdots,a_{l_1},b_1,b_2,\cdots,b_{l_2},-c_1,-c_2,\cdots,-c_{l_3})$. Other values in Q are set as 0. Finally, the search user sets $s=\sum_{h=1}^{l_2}b_h$. In the $Query$ phase, For a document F_j, if the corresponding $R_j>0$, we claim that F_j can satisfy the above matching rule.

Theorem 3: (Correctness) F_j satisfies the above matching rule with "OR", "AND" and "NO" if and only if the corresponding $R_j>0$.

Proof: Firstly, we proof the completeness. Since the weight of $w'''_i(i=1,2,\cdots,l_3)$ in the vector Q is $-c_i$ and $c_i>\sum_{k=1}^{l_1}a_k+\sum_{h=1}^{l_2}b_h$, if any corresponding value of w'''_i in P of F_j is 1, we can infer $P\cdot Q<0$ and $R_j=r\cdot(P\cdot Q-s)<0$. Therefore, if $R_j>0$, any of w'''_i is not in the keyword set of F_j, i.e., F_j satisfies the "NO" operation. Moreover, if $R_j>0$, then $r\cdot(P\cdot Q-s)=r\cdot(P\cdot Q-\sum_{h=1}^{l_2}b_h)>0$. Since $b_h>\sum_{k=1}^{l_1}a_k(h=1,2,\cdots,l_2)$, all corresponding values of w''_h in P have to be 1 and at least one corresponding value of $w'_k(k=1,2,\cdots,l_1)$ in P should be 1. Thus, F_j satisfies the "AND" and "OR" operations. Therefore, if $R_j>0$, the vector P satisfies the operations of "OR", "AND" and "NO".

Next, we show the soundness. If the vector P satisfies the operations of "OR", "AND" and "NO", i.e., at least one corresponding value of keyword w'_k in P is 1 (assume this keyword is $w'_\gamma(1\le\gamma\le l_1)$), all corresponding values of keywords w''_h in P are 1 and no corresponding value of keyword w'''_i in P is 1. Therefore, $R_j=r\cdot(P\cdot Q-s)\ge r\cdot(a_\gamma+b_1+b_2+\cdots+b_{l_2}-s)=r\cdot a_\gamma>0$.

Example. We present a concrete example to help understand *Theorem 3*. The example also illustrates the working process of FMS_II. Specifically, we assume that the keyword sets corresponding to the "OR", "AND" and "NO" operations are (w'_1,w'_2,w'_3), (w''_1,w''_2,w''_3) and (w'''_1,w'''_2), respectively. Thus, the matching rule can be represented as $(w'_1\vee w'_2\vee w'_3)\wedge(w''_1\wedge w''_2\wedge w''_3)\wedge(\neg w'''_1\wedge\neg w'''_2)$. we assume that the search weights (a_1,a_2,a_3), (b_1,b_2,b_3) and (c_1,c_2) for "OR", "AND" and "NO" are(1,5,8), (20,24,96) and (-500,-600), respectively. We firstly prove $R_j>0$ when F_j satisfies the matching rule. Specifically, assume that F_j satisfies the matching rule $w'_2\wedge(w''_1\wedge w''_2\wedge w''_3)\wedge(\neg w'''_1\wedge\neg w'''_2)$. Thus the corresponding values of vector P are $(0,1,0)$, $(1,1,1)$ and $(0,0)$, respectively. Thus, the result of $s=\sum_{h=1}^{3}b_h=20+24+96=140$,

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

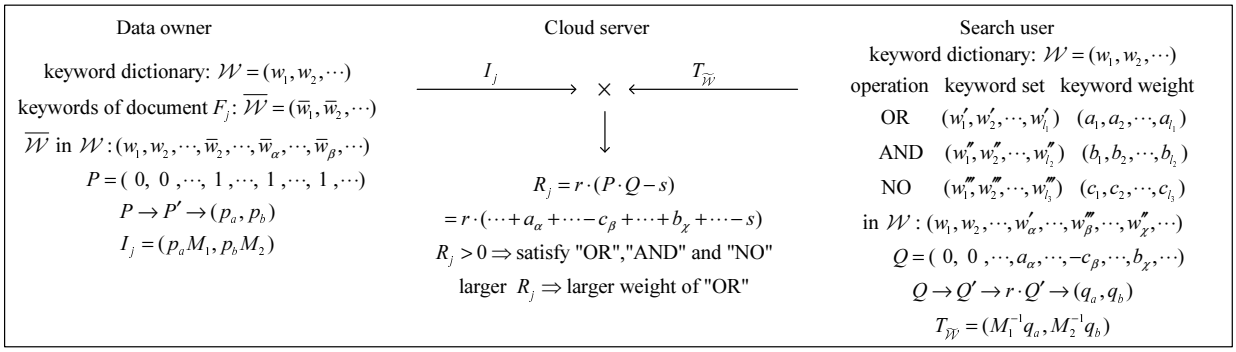

Fig. 3. Structure of the FMS_II

for arbitrary $r>0$, the result of R_j will be

$$\begin{aligned} R_j &= r\cdot(P\cdot Q-s)\\ &= r\cdot(a_2+b_1+b_2+b_3-s)\\ &= r\cdot(5+20+24+96-140)\\ &= 5r>0 \end{aligned} \tag{14}$$

From the above example, we can easily see that $R_j>0$ when F_j satisfies the matching rule. Next, we show that $R_j<0$ when F_j does not satisfy the matching rule. Especially, we assume that the "AND" operation does not satisfy the matching rule. Here, we set the first keyword does not match the rule, therefore the search keyword set of "AND" operations are $(0,1,1)$ instead of $(1,1,1)$. Thus, the result of R_i will be

$$\begin{aligned} R_j &= r\cdot(P\cdot Q-s)\\ &= r\cdot(a_2+b_2+b_3-s)\\ &= r\cdot(5+24+96-140)\\ &= -15r<0 \end{aligned} \tag{15}$$

Obviously, $R_j<0$ when F_j does not satisfy the matching rule.

5 ENHANCED SCHEME

In practice, apart from some common keywords, other keywords in dictionary are generally professional terms, and this part of the dictionary will rapidly increase when the dictionary becomes larger and more comprehensive. Simultaneously, the data owner's index will become longer, although many dimensions of keywords will never appear in her documents. That will cause redundant computation and communication overhead.

In this section, we further propose a Fine-grained Multi-keyword Search scheme supporting Classified Sub-dictionaries (FMSCS), which classifies the total dictionary as a common sub-dictionary and many professional sub-dictionaries. Our goal is to significantly reduce the computation and communication overhead. We have researched in a file set randomly chosen from the National Science Foundation (NSF) Research Awards Abstracts 1990-2003 [24]. As shown in Fig. 4, we classify the total dictionary to many sub-dictionaries such as common sub-dictionary, computer science sub-dictionary, mathematics sub-dictionary and physics sub-dictionary, etc. And the search process will only be some minor changes in *Initialization*.

Change of Initialization: Compared with the Basic Framework, in the enhanced scheme the data owner should firstly choose corresponding sub-dictionaries. Then her own dictionary can be combined as $\{f_1||Subdic_1||f_2||Subdic_2||\cdots\}$, where $Subdic_i$ represents all keywords contained in corresponding sub-dictionary and f_i is filling factor with random length which will be 0 string in the index, the filling factor is used to confuse length of the data owner's own dictionary and relative positions of sub-dictionaries. Then, the data owner and search user will use this dictionary to generate the index and trapdoor, respectively. Note that in an dictionary, two professional sub-dictionaries can even contain a same keyword, but only the first appeared keyword will be used to generate index and trapdoor, another will be set to 0 in the vector. And the secret key K will be formed as $(S,M_1,M_2,|f_1|,D_{ID_1},|f_2|,D_{ID_2},\cdots)$, where D_{ID_i} represents the identity of sub-dictionary and $|f_i|$ is the length of f_i. Other than these changes, the remaining phases (i.e., *Index building*, *Trapdoor generating* and *Query*) are same as the Basic Framework.

Dictionary Updating: In the searchable encryption schemes with dictionary, dictionary update is a challenge problem because it may cause to update massive indexes outsourced to the cloud server. In general dictionary-based search schemes, e.g., [13] and [14], the update of dictionary will lead to re-generation of all indexes. In our FMSCS schemes, when it needs to change the sub-dictionaries or add new sub-dictionaries, only the data owners who use the corresponding sub-dictionaries need to update their indexes, most other data owners do not need to do any update operations. Such dictionary update operations are particularly lightweight. In addition, Li et al. [9] utilize the dimension expansion technique to implement the efficient dictionary expansion. Such method can also be included into our dictionary updating process. And our scheme can even be more efficient than [9] since although [9] does not need to re-generate all indexes, but the corresponding extended operations on all indexes are necessary. In comparison, our schemes only need to extend the indexes of partial data owners.

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

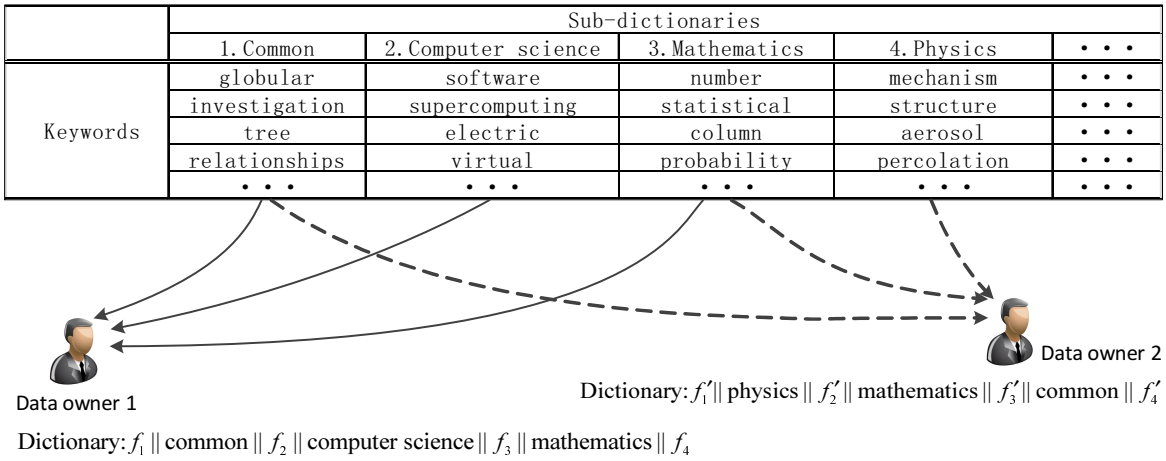

Fig. 4. Classified sub-dictionaries

6 SECURITY ANALYSIS

In this section, we analyze the main security properties of the proposed schemes. In particular, our analysis focuses on how the proposed schemes can achieve confidentiality of documents, privacy protection of index and trapdoor, and unlinkability of trapdoor. Other security features are not the focus of our concern.

6.1 Confidentiality of Documents

In our schemes, the outsourced documents are encrypted by the traditional symmetric encryption algorithm (e.g., AES). In addition, the secret key sk is generated by the data owner and sent to the search user through a secure channel. Since the AES encryption algorithm is secure [23], any entity cannot recover the encrypted documents without the secret key sk. Therefore, the confidentiality of encrypted documents can be achieved.

6.2 Privacy Protection of Index and Trapdoor

As shown in Section 4.1, both the index $I_j = (p_a M_1, p_b M_2)$ and the trapdoor $T_{\widetilde{\mathcal{W}}} = (M_1^{-1} q_a, M_2^{-1} q_b)$ are ciphertexts of vectors (P, Q). The secret key is $K = (S, M_1, M_2)$ in the FMS or $(S, M_1, M_2, |f_1|, D_{ID_1}, |f_2|, D_{ID_2}, \cdots)$ in the FMSCS, where S functions as a splitting indicator which splits P and Q into (p_a, p_b) and (q_a, q_b), respectively, two invertible matrices M_1 and M_2 are used to encrypt (p_a, p_b) and (q_a, q_b). The security of this encryption algorithm has been proved in the known ciphertext model [21]. Thus, the content of index and trapdoor cannot be identified. Therefore, privacy protection of index and trapdoor can be achieved.

6.3 Unlinkability of Trapdoor

To protect the security of search, the unlinkability of trapdoor should be achieved. Although the cloud server cannot directly recover the keywords, the linkability of trapdoor may cause leakage of privacy, e.g., the same keyword set may be searched many times, if the trapdoor generation function is deterministic, even though the cloud server cannot decrypt the trapdoors, it can deduce the relationship of keywords. We consider whether the trapdoor $T_{\widetilde{\mathcal{W}}} = (M_1^{-1} q_a, M_2^{-1} q_b)$ can be linked to the keywords. We prove our schemes can achieve the unlinkability of trapdoor in a strong threat model, i.e., known background model [6].

Known Background Model: In this model, the cloud server can possess the statistical information from a known comparable dataset which bears the similar nature to the targeting dataset.

TABLE 1
Structure of Q'

	$Q'[1] \cdots Q'[m]$	$Q'[m+1]$
FMS(CS)_I	$\cdots 0 \cdots d_i \cdots 0 \cdots d_j \cdots$	$-s$
FMS(CS)_II	$\cdots a_k \cdots b_h \cdots 0 \cdots c_i \cdots$	$-s$

As shown in Table 1, in our FMS(CS)_I, the trapdoor is constituted by two parts. The values of all dimensions $d_i (i = 1, 2, \cdots, l)$ are the super-increasing sequence randomly chosen by the search user (assume there are α possible sequences). And the $(m+1)$ dimension is $-s$ defined by the search user, where s is a positive random number. Assume the size of $-s$ is η_s bits, there are 2^{η_s} possible values for $-s$. Further, to generate $Q'' = r \cdot Q'$, Q' is multiplied by a positive random number r, there are 2^{η_r} possible values for r (if the search user chooses η_r-bit r). Finally, Q'' is split to (q_a, q_b) according the splitting indicator S. Specifically, if $S[i] = 0 (i = 1, 2, \cdots, m+1)$, the value of $Q''[i]$ will be randomly split into $q_a[i]$ and $q_b[i]$, assume in S the number of '0' is μ, and each dimension of q_a and q_b is η_q bits. Note that η_s, η_r, μ and η_q are independent of each other. Then in our FMS(CS)_I, we can compute the probability that two trapdoors are the same as follows:

$$P_1 = \frac{1}{\alpha \cdot 2^{\eta_s} \cdot 2^{\eta_r} \cdot (2^{\eta_q})^{\mu}} = \frac{1}{\alpha \cdot 2^{\eta_s + \eta_r + \mu \eta_q}} \qquad (16)$$

Therefore, the larger α, η_s, η_r, μ and η_q can achieve the stronger security, e.g., if we choose 1024-bit r, then the

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

probability $P_1 < 1/2^{1024}$. As a result, the probability that two trapdoors are the same is negligible.

And in the FMS(CS)_II, because $-s = -\sum_{h=1}^{l_2} b_h$, its value depends on the weight sequence $(a_1, a_2, \cdots, a_{l_1}, b_1, b_2, \cdots, b_{l_2}, c_1, c_2, \cdots, c_{l_3})$. Assume the number of different sequences is denoted as β, then we can compute:

$$P_2 = \frac{1}{\beta \cdot 2^{\eta_r} \cdot (2^{\eta_q})^{\mu}} = \frac{1}{\beta \cdot 2^{\eta_r + \mu\eta_q}} \tag{17}$$

Similarly, in the FMS(CS)_II and the FMS(CS)_III, the probability that two trapdoors are the same is negligible. Therefore, in our schemes, the unlinkability of trapdoor can be achieved.

In summary, we present the comparison results of security level in Table 2, where I and II represent FMS(CS)_I and FMS(CS)_II, respectively. It can be seen that all schemes can achieve confidentiality of documents and privacy protection of index and trapdoor, but the OPE schemes [11], [25] cannot achieve the unlinkability of trapdoor very well because of the similarity relevance mentioned in [14].

TABLE 2
Comparison of Security Level

	[11], [25]	[6], [13], [14]	I	II
Confidentiality	√	√	√	√
Privacy protection	√	√	√	√
Unlinkability		√	√	√

Discussions: In MRSE [6], the values of $P \cdot Q$ are equal to the number of matching keywords, which suffers scale analysis attack when the cloud server is powerful and has knowledge of some background information. To solve this problem, it extends the index and inserts a random number ε_j which follows a normal distribution and can confuse the values of $P \cdot Q$. Thus, enhanced MRSE can resist scale analysis attack. However, the introduction of ε_j causes precision decrease of the returned results. There is a trade-off between precision and security in MRSE. In comparison, our schemes do not suffer the scale analysis attack. Because the values of $P \cdot Q$ in our schemes do not disclose any information due to the randomly selected sequences mentioned in Section 4.2 and Section 4.3. Therefore, our proposal can achieve the security without sacrificing precision.

7 PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed schemes using simulations, and compare the performance with that of existing proposals in [6], [13], [14]. We apply a real-world dataset from the National Science Foundation Research Awards Abstracts 1990-2003 [24], in which we random select multiple documents and conduct real-world experiments on an Intel Core i5 3.2 GHz system.

7.1 Functionality

We compare functionalities between [6], [13], [14] and our schemes in Table 3, where I and II represent FMS(CS)_I and FMS(CS)_II, respectively.

MRSE [6] can achieve multi-keyword search and coordinate matching using secure kNN computation scheme. And [13] and [14] considers the relevance scores of keywords. Compared with the other schemes, our FMS(CS)_I considers both the relevance scores and the preference factors of keywords. Note that if the search user sets all relevance scores and preference factors of keywords as the same, the FMS(CS)_I degrades to MRSE and the coordinate matching can be achieved. And in the FMS(CS)_II, if the search user sets all preference factors of "OR" operation keywords as the same, the FMS(CS)_II can also achieve the coordinate matching of "OR" operation keywords. Particularly, the FMS(CS)_II achieves some fine-grained operations of keyword search, i.e., "AND", "OR" and "NO" operations in Google Search, which are definitely practical and significantly enhance the functionalities of encrypted keyword search.

TABLE 3
Comparison of Functionalities

	[6]	[13]	[14]	I	II
Multi-keyword search	√	√	√	√	√
Coordinate matching	√	√	√	√	√
Relevance score		√	√	√	
Preference factor				√	√
AND OR NO operations					√

7.2 Query Complexity

In the FMS(CS)_II, we can implement "OR", "AND" and "NO" operations by defining appropriate weights of keywords, this scheme provides a more fine-grained search than [6], [13] and [14]. If the keywords to perform "OR", "AND" and "NO" operations are $(w'_1, w'_2, \cdots, w'_{l_1})$, $(w''_1, w''_2, \cdots, w''_{l_2})$ and $(w'''_1, w'''_2, \cdots, w'''_{l_3})$, respectively. Our FMS(CS)_II can complete the search with only one query, however, in [6], [13] and [14], they would complete the search through the following steps:

- For the "OR" operation of l_1 keywords, they need only one query $Query(w'_1, w'_2, \cdots, w'_{l_1})$ to return a collection of documents with the most matching keywords (i.e., coordinate matching), which can be denoted as $X = Query(w'_1, w'_2, \cdots, w'_{l_1})$.
- For the "AND" operation of l_2 keywords, [6], [13] and [14] cannot generate a query for multiple keywords to achieve the "AND" operation. Therefore, after costing l_2 queries $Query(w''_i)(i = 1, 2, \cdots, l_2)$, they can do the "AND" operation, and the corresponding document set can be denoted as $Y = Query(w''_1) \bigcap Query(w''_2) \bigcap \cdots \bigcap Query(w''_{l_2})$.
- For the "NO" operation of l_3 keywords, they need l_3 queries $Query(w'''_i)(i = 1, 2, \cdots, l_3)$, firstly. Then, the document set of the "NO" operation can be denoted as $Z = \overline{Query(w'''_1)} \bigcap \overline{Query(w'''_2)} \bigcap \cdots \bigcap \overline{Query(w'''_{l_3})}$.
- Finally, the document collection achieved "OR", "AND" and "NO" operations can be represented as $X \bigcap Y \bigcap Z$.

As shown in Fig. 5a, 5b and 5c, to achieve these operations, the FMS(CS)_II can outperform the existing proposals with less queries generated.

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

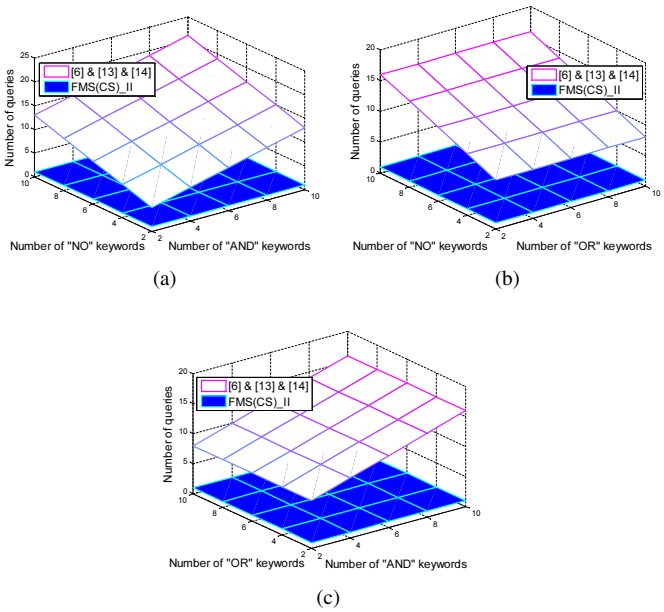

Fig. 5. Time for building index. (a) Number of queries for the different number of "AND" and "NO" keywords with the same number of "OR" keywords, $l_1 = 5$. (b) Number of queries for the different number of "OR" and "NO" keywords with the same number of "AND" keywords, $l_2 = 5$. (c) Number of queries for the different number of "AND" and "OR" keywords with the same number of "NO" keywords, $l_3 = 5$.

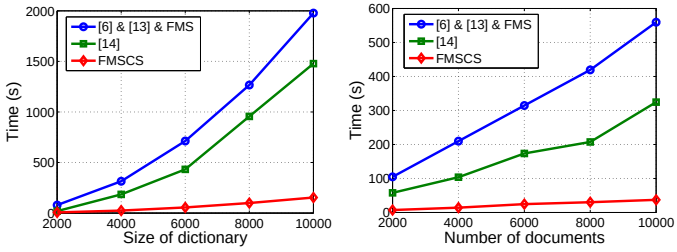

Fig. 6. Time for building index. (a) For the different size of dictionary with the same number of documents, N=6000. (b) For the different number of documents with the same size of dictionary, $|\mathcal{W}| = 4000$.

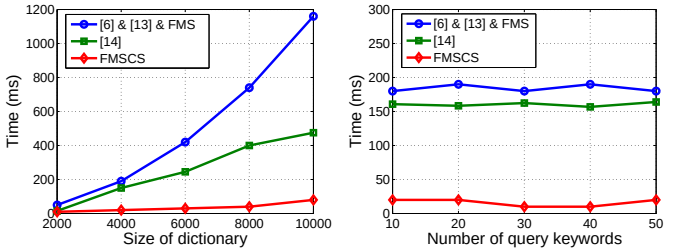

Fig. 7. Time for generating trapdoor. (a) For the different size of dictionary with the same number of query keywords, $|\widetilde{\mathcal{W}}|$=20. (b) For the different number of query keywords with the same size of dictionary, $|\mathcal{W}| = 4000$.

7.3 Efficiency

7.3.1 Computation overhead

In order to easily demonstrate our scheme computation overhead, we analysis our scheme from each phase.

Index building. Note that the *Index building* phase of [6] is the same as our FMS_II scheme, without calculating the relevance score. And the *Index building* phase of the FMS_I is the same as [13], containing the relevance score computing. Compared with the FMS_I, the FMS_II does not need to calculate the relevance score. And compared with the computation cost of building index, the cost of calculating the relevance score is negligible, we do not distinguish them. Moreover, in our enhanced schemes (FMSCS), we divide the total dictionary into 1 common sub-dictionary and 20 professional sub-dictionaries (assume each data owner averagely chooses 1 common sub-dictionary and 3 professional sub-dictionaries to generate the index). As shown in Fig. 6, we can see the time for building index is dominated by both the size of dictionary and the number of documents. And compared with [6], [13], [14] and our FMS schemes, the FMSCS schemes largely reduce the computation overhead.

Trapdoor generating. In *Trapdoor generating* phase, [6] and [13] firstly creates a vector according to the search keyword set $\widetilde{\mathcal{W}}$, then encrypts the vector by the secure kNN computation scheme. And [14] also generates a vector and uses homomorphic encryption to encrypt each dimension. In comparison, our FMS_I and FMS_II schemes should firstly generate a super-increasing sequence and a weight sequence, respectively. But actually, we can pre-select a corresponding sequence for each scheme, it can also achieve search process and privacy. Because even if the vectors are the same for multiple queries, the trapdoors will be not the same due to the security of kNN computation scheme. Therefore, the computation cost of [6], [13] and all FMS schemes in *Trapdoor generating* phase are the same. As shown in Fig. 7, the time for generating trapdoor is dominated by the size of dictionary, instead of the number of query keywords. Hence, our FMSCS schemes are also very efficient in *Trapdoor generating* phase.

Query. As [6], [13] and the FMS all adopt the secure kNN computation scheme, the time for query is the same. The computation overhead in *Query* phase, as shown in Fig. 8, is greatly affected by the size of dictionary and the number of documents, and almost has no relation to the number of query keywords. Further we can see, our FMSCS schemes significantly reduce the computation cost in *Query* phase. As [14] needs to encrypt each dimension of index/trapdoor using full homomorphic encryption, its index/trapdoor size is enormous. Note that, in *Trapdoor generating* and *Query* phases, the computation overheads are not affected by the number of query keywords. Thus our FMS and FMSCS schemes are more efficient compared with some multiple-keyword search schemes [26], [27], as their cost is linear with the number of query 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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

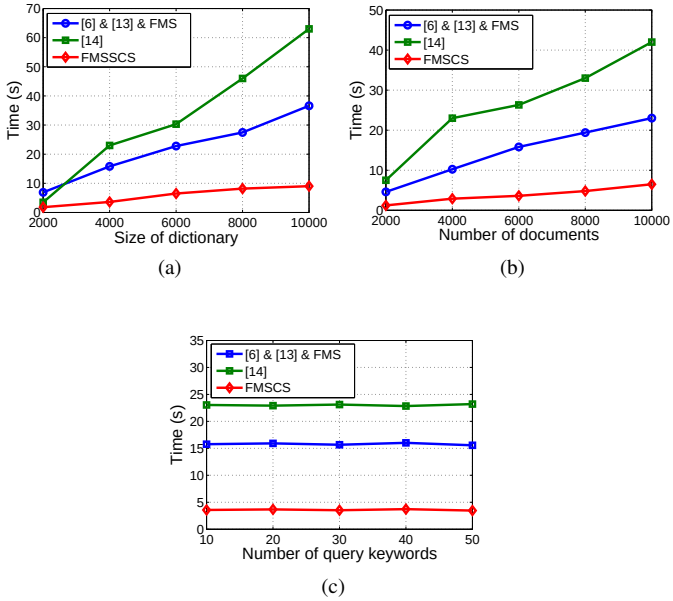

Fig. 8. Time for query. (a) For the different size of dictionary with the same number of documents and number of search keywords, $N = 6000, |\widetilde{\mathcal{W}}| = 20$. (b) For the different number of documents with the same size of dictionary and number of search keywords, $|\mathcal{W}| = 4000, |\widetilde{\mathcal{W}}| = 20$. (c) For the different number of search keyword with the same size of dictionary and number of documents, $N = 6000, |\mathcal{W}| = 4000$.

7.3.2 Storage overhead

As shown in Table 4, we provide a comparison of storage overhead among several schemes. Specifically, we evaluate the storage overhead from three parts: the data owner, the search user and the cloud server.

According to Table 4, in the FMS, the FMSCS as well as schemes of [6] and [13], the storage overhead of the data owner are the same. In these schemes, the data owner preserves her secret key $K = (S, M_1, M_2)$ and symmetric key sk locally, where S is an $(m+1)$-dimensional vector, M_1 and M_2 are $(m+1)\times(m+1)$ invertible matrices. All elements in S, M_1 and M_2 are the float number. Since the size of a float number is 4 bytes, the size of K is $4\cdot(m+1)+8\cdot(m+1)^2$ bytes. We assume that the size of sk is S_{sk} that is a constant. Thus, the total size of storage overhead is $4\cdot(m+1)+8\cdot(m+1)^2+S_{sk}$ bytes. However, in [14], the storage overhead of data owner is $\lambda^5/8$ bytes, where the λ is the secure parameter. The storage overhead is 4GB when we choose $\lambda = 128$, which is popular in a full homomorphic encryption scheme. However, the storage overhead of the FMS and the FMSCS are almost 763MB when we choose $m = 10000$, which is large enough for a search scheme. Therefore, the FMS and the FMSCS are more efficient than scheme in [14] in terms of the storage overhead of the data owner.

As shown in Table 4, a search user in the FMS, the FMSCS as well as the schemes of [6] and [13] preserves the secret key $K = (S, M_1, M_2)$ and the symmetric key sk locally. Therefore, the total storage overhead is $4(m+1)+8(m+1)^2+S_{sk}$ bytes. However, in [14], the storage overhead is $\lambda^5/8+\lambda^2/8$ bytes. The storage overhead is 4GB when we choose $\lambda = 128$, which is popular in a full homomorphic encryption scheme. However, the storage overhead of the FMS and the FMSCS are almost 763MB when we choose $m = 10000$, which is large enough for a search scheme. Therefore, the FMS and the FMSCS are more efficient than scheme in [14] in terms of the storage overhead of the search user.

The cloud server preserves the encrypted documents and the indexes. The size of encrypted documents in all schemes are the same, i.e., $N\cdot D_s$. For the indexes, in the FMS and schemes in [6] and [13], the storage overhead are $8\cdot(m+1)\cdot N$ bytes. In the FMSCS, the storage overhead is $8\cdot\varepsilon\cdot(m+1)\cdot N$ bytes, where $0 < \varepsilon < 1$. When $m = 1000$ and $N = 10000$ which are large enough for a search scheme, the storage overhead of indexes is about 132MB in the FMSCS. And in schemes of [6] and [13] as well as the FMS, the size of indexes is 760MB with the same conditions. In scheme in [14], the storage overhead of indexes is $N\cdot D_s + m\cdot N\cdot(\lambda/8)^5$ bytes, it is 4GB when we choose $\lambda = 128$, which is popular in a full homomorphic encryption scheme. Therefore, the FMS and the FMSCS are more efficient than scheme in [14] in terms of the storage overhead of the cloud server.

7.3.3 Communication overhead

As shown in Table 5, we provide a comparison of communication overhead among several schemes. Specifically, we consider the communication overhead from three parts: the communication between the data owner and the cloud server (abbreviated as **D-C**), the communication between the search user and the cloud server (abbreviated as **C-S**) and the communication between the data owner and the search user (abbreviated as **D-S**).

D-C. In the FMS as well as schemes of [6] and [13], the data owner needs to send information to cloud server in the form of $C_j||FID_j||I_j$ $(j = 1, 2, \cdots, N)$, where the C_j represents the encrypted documents, FID_j represents the identity of the document and I_j represents the index. We assume that the average size of documents is D_s, thus the size of documents is $N\cdot D_s$. We assume the encrypted documents identity FID is a 10-byte string. Thus, the total size of the identity FID is $10\cdot N$ bytes. The index $I_j = (p_a M_1, p_b M_2)$ contains two (m+1)-dimensional vectors. Each dimension is a float number (the size of each float is 4 bytes). Thus, the total size of index is $8\cdot(m+1)\cdot N$ bytes. Therefore, the total size of communication overhead is $8\cdot(m+1)\cdot N+10\cdot N+N\cdot D_s$ bytes. In the FMSCS, the total size of communication overhead is $8\cdot\varepsilon\cdot(m+1)\cdot N+10\cdot N+N\cdot D_s$ bytes. If we choose the ε as 0.2, the size of index is $1.6\cdot(m+1)\cdot N$ bytes, and the total size of communication of FMSCS is $1.6\cdot(m+1)\cdot N+10\cdot N+D_s\cdot N$ bytes. However, in [14], the communication overhead is $N\cdot D_s+m\cdot N\cdot\lambda^5/8$ bytes, where λ is the secure parameter. If we choose $\lambda = 128$ which is popular in a full homomorphic encryption scheme and $m = 1000$ and $N = 10000$ which are large enough for a search scheme, the FMS and the FMSCS are more efficient than scheme in [14] in terms of the communication overhead of **D-C**.

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

TABLE 4
Comparison of Storage Overhead (Bytes). (m represents the size of dictionary; N represents the number of documents; D_s represents the average size of each encrypted document; λ represents the secure parameter; ε represents the decrease rate of dictionary by using our classified sub-dictionaries technology; S_{sk} represents the size of symmetric key.)

	[14]	[6], [13] and FMS	FMSCS
Data Owner	$\lambda^5/8$	$4\cdot(m+1)+8\cdot(m+1)^2+S_{sk}$	$4\cdot(m+1)+8\cdot(m+1)^2+S_{sk}$
Search User	$\lambda^5/8+\lambda^2/8$	$4\cdot(m+1)+8\cdot(m+1)^2+S_{sk}$	$4\cdot(m+1)+8\cdot(m+1)^2+S_{sk}$
Cloud Server	$N\cdot D_s+m\cdot N\cdot\lambda^5/8$	$N\cdot D_s+8\cdot(m+1)\cdot N$	$N\cdot D_s+8\cdot\varepsilon\cdot(m+1)\cdot N$

TABLE 5
Comparison of Communication Overhead (Bytes). (m represents the size of dictionary; N represents the number of documents; D_s represents the average size of each encrypted document; T represents the number of returned documents; λ represents the secure parameter; ε represents the decrease rate of dictionary by using our classified sub-dictionaries technology; S_{sk} represents the size of symmetric key.)

	[14]	[6], [13] and FMS	FMSCS
D-C	$N\cdot D_s+m\cdot N\cdot\lambda^5/8$	$8\cdot(m+1)\cdot N+10\cdot N+N\cdot D_s$	$8\cdot\varepsilon\cdot(m+1)\cdot N+10\cdot N+N\cdot D_s$
C-S	$m\cdot\lambda^5/8+T\cdot D_s$	$8\cdot(m+1)+T\cdot D_s$	$8\cdot\varepsilon\cdot(m+1)+T\cdot D_s$
D-S	$\lambda^5/8+\lambda^2/8$	$4\cdot(m+1)+8\cdot(m+1)^2+S_{sk}$	$4\cdot(m+1)+8\cdot(m+1)^2+S_{sk}$

C-S. The **C-S** consists of two phases: *Query* and *Results returning*. In the *Query* phase, a search user in the FMS as well as the schemes in [6] and [13] sends the trapdoor to the cloud server in the form of $T_{\widetilde{\mathcal{W}}}=(M_1^{-1}q_a, M_2^{-1}q_b)$, which contains two (m+1)-dimensional vectors. Thus, the communication overhead is $8\cdot(m+1)$ bytes. In the FMSCS, the communication overhead is $8\cdot\varepsilon\cdot(m+1)(0<\varepsilon<1)$ bytes. In the *Results returning* phase, the cloud server sends the corresponding result to the search user. The communication overhead of **C-S** increases along with the number of returned documents at this point. We assume that the number of the returned documents is T, thus, the total communication overhead of cloud server to search user is $T\cdot D_s$ bytes. Therefore, the total communication overhead of **C-S** is $8\cdot m+T\cdot D_s$ bytes. In the FMS as well as the schemes in [6] and [13], the total communication overhead of **C-S** is $8\cdot\varepsilon\cdot(m+1)+T\cdot D_s$ bytes. In [14], the total communication overhead of **C-S** is $m\cdot\lambda^5/8+T\cdot D_s$ bytes. If we choose $\lambda=128$ which is popular in a full homomorphic encryption scheme and $m=1000$ and $N=10000$ which are large enough for a search scheme, the FMS and the FMSCS are more efficient than scheme in [14] in terms of the communication overhead of **C-S**.

D-S. From table 5, we can see that the communication overhead of the FMS, the FMSCS as well as schemes in [6] and [13] are the same. In the *Initialization* phase, the data owner sends the secret key $K=(S,M_1,M_2)$ and symmetric key sk to the search user, where S is an $(m+1)$-dimensional vector, M_1 and M_2 are $(m+1)\times(m+1)$ invertible matrices. Thus, the size of the secret key K is $4\cdot(m+1)+8\cdot(m+1)^2$ bytes. Therefore, the total size of communication overhead is $4\cdot(m+1)+8\cdot(m+1)^2+S_{sk}$ bytes, where the S_{sk} represents the size of symmetric key. However, the communication overhead of scheme in [14] is $\lambda^5/8+\lambda^2/8$ bytes. The communication overhead is 4GB when we choose $\lambda=128$, which is popular in a full homomorphic encryption scheme. However, the communication overhead of the FMS and the FMSCS are almost 763MB when we choose $m=10000$, which is large enough for a search scheme. Therefore, the FMS and the FMSCS are more efficient than scheme in [14] in terms of the communication overhead of **D-S**.

8 RELATED WORK

There are mainly two types of searchable encryption in literature, Searchable Public-key Encryption (SPE) and Searchable Symmetric Encryption (SSE).

8.1 SPE

SPE is first proposed by Boneh et al. [28], which supports single keyword search on encrypted data but the computation overhead is heavy. In the framework of SPE, Boneh et al. [27] propose conjunctive, subset, and range queries on encrypted data. Hwang et al. [29] propose a conjunctive keyword scheme which supports multi-keyword search. Zhang et al. [17] propose an efficient public key encryption with conjunctive-subset keywords search. However, these conjunctive keywords schemes can only return the results which match all the keywords simultaneously, and cannot rank the returned results. Qin et al. [30] propose a ranked query scheme which uses a mask matrix to achieve cost-effectiveness. Yu et al. [14] propose a multi-keyword top-k retrieval scheme with fully homomorphic encryption, which can return ranked results and achieve high security. In general, although SPE allows more expressive queries than SSE [13], it is less efficient, and therefore we adopt SPE in the work.

8.2 SSE

The concept of SSE is first developed by Song et al. [8]. Wang et al. [25] develop the ranked keyword search scheme, which considers the relevance score of a keyword. However, the above schemes cannot efficiently support multi-keyword search which is widely used to provide the better experience

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

to the search user. Later, Sun et al. [13] propose a multi-keyword search scheme which considers the relevance scores of keywords, and it can achieve efficient query by utilizing the multidimensional tree technique. A widely adopted multi-keyword search approach is multi-keyword ranked search (MRSE) [6]. This approach can return the ranked results of searching according to the number of matching keywords. Li et al. [10] utilize the relevance score and k-nearest neighbor techniques to develop an efficient multi-keyword search scheme that can return the ranked search results based on the accuracy. Within this framework, they leverage an efficient index to further improve the search efficiency, and adopt the blind storage system to conceal access pattern of the search user. Li et al. [19] also propose an authorized and ranked multi-keyword search scheme (ARMS) over encrypted cloud data by leveraging the ciphertext policy attribute-based encryption (CP-ABE) and SSE techniques. Security analysis demonstrates that the proposed ARMS scheme can achieve collusion resistance. In this paper, we propose FMS(CS) schemes which not only support multi-keyword search over encrypted data, but also achieve the fine-grained keyword search with the function to investigate the relevance scores and the preference factors of keywords and, more importantly, the logical rule of keywords. In addition, with the classified sub-dictionaries, our proposal is efficient in terms of index building, trapdoor generating and query.

9 Conclusion

In this paper, we have investigated on the fine-grained multi-keyword search (FMS) issue over encrypted cloud data, and proposed two FMS schemes. The FMS_I includes both the relevance scores and the preference factors of keywords to enhance more precise search and better users' experience, respectively. The FMS_II achieves secure and efficient search with practical functionality, i.e., "AND", "OR" and "NO" operations of keywords. Furthermore, we have proposed the enhanced schemes supporting classified sub-dictionaries (FMSCS) to improve efficiency.

For the future work, we intend to further extend the proposal to consider the extensibility of the file set and the multi-user cloud environments. Towards this direction, we have made some preliminary results on the extensibility [5] and the multi-user cloud environments [19]. Another interesting topic is to develop the highly scalable searchable encryption to enable efficient search on large practical databases.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grants 61472065, 61350110238, 61103207, U1233108, U1333127, and 61272525, the International Science and Technology Cooperation and Exchange Program of Sichuan Province, China under Grant 2014HH0029, China Postdoctoral Science Foundation funded project under Grant 2014M552336, and State Key Laboratory of Information Security Open Foundation under Grant 2015-MS-02.

References

[1] H. Liang, L. X. Cai, D. Huang, X. Shen, and D. Peng, "An smdp-based service model for interdomain resource allocation in mobile cloud networks," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 5, pp. 2222–2232, 2012.

[2] M. M. Mahmoud and X. Shen, "A cloud-based scheme for protecting source-location privacy against hotspot-locating attack in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 10, pp. 1805–1818, 2012.

[3] Q. Shen, X. Liang, X. Shen, X. Lin, and H. Luo, "Exploiting geo-distributed clouds for e-health monitoring system with minimum service delay and privacy preservation," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 2, pp. 430–439, 2014.

[4] T. Jung, X. Mao, X. Li, S.-J. Tang, W. Gong, and L. Zhang, "Privacy-preserving data aggregation without secure channel: multivariate polynomial evaluation," in *Proceedings of INFOCOM*. IEEE, 2013, pp. 2634–2642.

[5] Y. Yang, H. Li, W. Liu, H. Yang, and M. Wen, "Secure dynamic searchable symmetric encryption with constant document update cost," in *Proceedings of GLOBCOM*. IEEE, 2014, to appear.

[6] 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.

[7] https://support.google.com/websearch/answer/173733?hl=en.

[8] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of S&P*. IEEE, 2000, pp. 44–55.

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

[10] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. Shen, "Enabling efficient multi-keyword ranked search over encrypted cloud data through blind storage," *IEEE Transactions on Emerging Topics in Computing*, 2014, DOI10.1109/TETC.2014.2371239.

[11] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proceedings of ICDCS*. IEEE, 2010, pp. 253–262.

[12] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill, "Order-preserving symmetric encryption," in *Advances in Cryptology-EUROCRYPT*. Springer, 2009, pp. 224–241.

[13] 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. DOI: 10.1109/TPDS.2013.282, 2013.

[14] J. Yu, P. Lu, Y. Zhu, G. Xue, and M. Li, "Towards secure multi-keyword top-k retrieval over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 4, pp. 239–250, 2013.

[15] A. Arvanitis and G. Koutrika, "Towards preference-aware relational databases," in *International Conference on Data Engineering (ICDE)*. IEEE, 2012, pp. 426–437.

[16] G. Koutrika, E. Pitoura, and K. Stefanidis, "Preference-based query personalization," in *Advanced Query Processing*. Springer, 2013, pp. 57–81.

[17] B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 262–267, 2011.

[18] D. Stinson, *Cryptography: theory and practice*. CRC press, 2006.

[19] H. Li, D. Liu, K. Jia, and X. Lin, "Achieving authorized and ranked multi-keyword search over encrypted cloud data," in *Proceedings of ICC*. IEEE, 2015, to appear.

[20] S. Zerr, E. Demidova, D. Olmedilla, W. Nejdl, M. Winslett, and S. Mitra, "Zerber: r-confidential indexing for distributed documents," in *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*. ACM, 2008, pp. 287–298.

[21] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of SIGMOD International Conference on Management of data*. ACM, 2009, pp. 139–152.

[22] J. Zobel and A. Moffat, "Exploring the similarity space," in *ACM SIGIR Forum*, vol. 32, no. 1. ACM, 1998, pp. 18–34.

[23] N. Ferguson, R. Schroeppel, and D. Whiting, "A simple algebraic representation of rijndael," in *Selected Areas in Cryptography*. Springer, 2001, pp. 103–111.

[24] http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html.

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/TDSC.2015.2406704, IEEE Transactions on Dependable and Secure Computing

[25] 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.
[26] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and Network Security*. Springer, 2004, pp. 31–45.
[27] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of cryptography*. Springer, 2007, pp. 535–554.
[28] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology–Eurocrypt*. Springer, 2004, pp. 506–522.
[29] Y. Hwang and P. Lee, "Zpublic key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proceeding of Pairing*. Springer, 2007, pp. 2–22.
[30] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Efficient information retrieval for ranked queries in cost-effective cloud environments," in *Proceedings of INFOCOM*. IEEE, 2012, pp. 2581–2585.

Hongwei Li is an Associate Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, China. He received the PhD degree in computer software and theory from University of Electronic Science and Technology of China, China in 2008. He has worked as a Post-Doctoral Fellow in Department of Electrical and Computer Engineering at University of Waterloo for one year until Oct. 2012. His research interests include network security, applied cryptography, and trusted computing. Dr. Li serves as the Associate Editor of Peer-to-Peer Networking and Applications, the Guest Editor for Peer to-Peer Networking and Applications Special Issue on Security and Privacy of P2P Networks in Emerging Smart City. He also serves on the technical program committees for many international conferences such as IEEE INFOCOM, IEEE ICC, IEEE GLOBECOM, IEEE WCNC, IEEE SmartGridComm, BODYNETS and IEEE DASC. He is a member of IEEE, a member of China Computer Federation and a member of China Association for Cryptologic Research.

Yi Yang received his B.S. degree in Network Engineering from Tianjin University of Science and Technology (TUST) in 2012. Currently, he is a master student at the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), China. He serves as the reviewer of Peer-to-Peer Networking and Application, IEEE INFOCOM, IEEE ICC, IEEE GLOBECOM, IEEE ICCC, etc. His research interests include cryptography, and the secure smart grid.

Tom H. Luan received the B.Sc. degree from Xian Jiaotong University, China, in 2004, the M.Phil. degree from Hong Kong University of Science and Technology, Hong Kong, China, in 2007, and Ph.D. degrees from the University of Waterloo, Canada, in 2012. Since December 2013, he has been the Lecturer in Mobile and Apps at the School of Information Technology, Deakin University, Melbourne Burwood, Australia. His research mainly focuses on vehicular networking, wireless content distribution, peer-to-peer networking and mobile cloud computing.

Xiaohui Liang received the B.Sc. degree in Computer Science and Engineering and the M.Sc. degree in Computer Software and Theory from Shanghai Jiao Tong University (SJTU), China, in 2006 and 2009, respectively. He is currently working toward a Ph.D. degree in the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include applied cryptography, and security and privacy issues for e-healthcare system, cloud computing, mobile social networks, and smart grid.

Liang Zhou is a professor with the National Key Lab of Science and Technology on Communication at University of Electronic Science and Technology of China, China. His current research interests include error control coding, secure communication and cryptography.

Xuemin (Sherman) Shen is a Professor and University Research Chair, Department of Electrical and Computer Engineering, University of Waterloo, Canada. He was the Associate Chair for Graduate Studies from 2004 to 2008. Dr. Shen's research focuses on resource management in interconnected wireless/wired networks, wireless network security, vehicular ad hoc and sensor networks. Dr. Shen served as the Technical Program Committee Chair for IEEE VTC'10 Fall and IEEE Globecom'07. He also serves/served as the Editor-in-Chief for IEEE Network, Peer-to-Peer Networking and Application, and IET Communications; a Founding Area Editor for IEEE Transactions on Wireless Communications; an Associate Editor for IEEE Transactions on Vehicular Technology, Computer Networks. Dr. Shen is a registered Professional Engineer of Ontario, Canada, an IEEE Fellow, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, and a Distinguished Lecturer of IEEE Vehicular Technology Society and Communications Society.