Received 30 June 2014; revised 12 October 2014; accepted 12 November 2014. Date of publication 11 December, 2014;
date of current version 6 March, 2015.

Digital Object Identifier 10.1109/TETC.2014.2371239

Enabling Efficient Multi-Keyword Ranked Search Over Encrypted Mobile Cloud Data Through Blind Storage

HONGWEI LI[1] (Member, IEEE), DONGXIAO LIU[1] (Student Member, IEEE), YUANSHUN DAI[1] (Member, IEEE), TOM H. LUAN[2] (Member, IEEE), AND XUEMIN (SHERMAN) SHEN[3] (Fellow, IEEE)

[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China
[2]School of Information Technology, Deakin University, Melbourne, VIC 3125, Australia
[3]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada

CORRESPONDING AUTHOR: H. LI (hongwei.uestc@gmail.com)

This work was supported in part by the International Science and Technology Cooperation and Exchange Program of Sichuan Province, China, under Grant 2014HH0029, in part by the China Post-Doctoral Science Foundation under Grant 2014M552336, and in part by the National Natural Science Foundation of China under Grant 61472065, Grant 61350110238, Grant U1233108, Grant U1333127, Grant 61272525, and 61472065,.

ABSTRACT In mobile cloud computing, a fundamental application is to outsource the mobile data to external cloud servers for scalable data storage. The outsourced data, however, need to be encrypted due to the privacy and confidentiality concerns of their owner. This results in the distinguished difficulties on the accurate search over the encrypted mobile cloud data. To tackle this issue, in this paper, we develop the searchable encryption for multi-keyword ranked search over the storage data. Specifically, by considering the large number of outsourced documents (data) in the cloud, we 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, we leverage an efficient index to further improve the search efficiency, and adopt the blind storage system to conceal access pattern of the search user. Security analysis demonstrates that our scheme can achieve confidentiality of documents and index, trapdoor privacy, trapdoor unlinkability, and concealing access pattern of the search user. Finally, using extensive simulations, we show that our proposal can achieve much improved efficiency in terms of search functionality and search time compared with the existing proposals.

INDEX TERMS Cloud computing, searchable encryption, multi-keyword ranked search, blind storage, access pattern.

I. INTRODUCTION

Mobile cloud computing [1]–[4] gets rid of the hardware limitation of mobile devices by exploring the scalable and virtualized cloud storage and computing resources, and accordingly is able to provide much more powerful and scalable mobile services to users. In mobile cloud computing, mobile users typically outsource their data to external cloud servers, e.g., iCloud, to enjoy a stable, low-cost and scalable way for data storage and access. However, as outsourced data typically contain sensitive privacy information, such as personal photos, emails, etc., which would lead to severe confidentiality and privacy violations [5], if without efficient protections. It is therefore necessary to encrypt the sensitive data before outsourcing them to the cloud. The data encryption, however, would result in salient difficulties when other users need to access interested data with search, due to the difficulties of search over encrypted data. This fundamental issue in mobile cloud computing accordingly motivates an extensive body of research in the recent years on the investigation of searchable encryption technique to achieve efficient searching over outsourced encrypted data [6]–[9].

A collection of research works have recently been developed on the topic of multi-keyword search over encrypted data. Cash et al. [10] propose a symmetric searchable encryption scheme which achieves high efficiency for large

2168-6750 © 2014 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

databases with modest scarification on security guarantees. Cao et al. [11] propose a multi-keyword search scheme supporting result ranking by adopting k-nearest neighbors (kNN) technique [12]. Naveed et.al. [13] propose a dynamic searchable encryption scheme through blind storage to conceal access pattern of the search user.

In order to meet the practical search requirements, search over encrypted data should support the following three functions. First, the searchable encryption schemes should support multi-keyword search, and provide the same user experience as searching in Google search with different keywords; single-keyword search is far from satisfactory by only returning very limited and inaccurate search results. Second, to quickly identify most relevant results, the search user would typically prefer cloud servers to sort the returned search results in a relevance-based order [14] ranked by the relevance of the search request to the documents. In addition, showing the ranked search to users can also eliminate the unnecessary network traffic by only sending back the most relevant results from cloud to search users. Third, as for the search efficiency, since the number of the documents contained in a database could be extraordinarily large, searchable encryption schemes should be efficient to quickly respond to the search requests with minimum delays.

In contrast to the theoretical benefits, most of the existing proposals, however, fail to offer sufficient insights towards the construction of full functioned searchable encryption as described above. As an effort towards the issue, in this paper, we propose an efficient multi-keyword ranked search (EMRS) scheme over encrypted mobile cloud data through blind storage. Our main contributions can be summarized as follows:

- We introduce a relevance score in searchable encryption to achieve multi-keyword ranked search over the encrypted mobile cloud data. In addition to that, we construct an efficient index to improve the search efficiency.
- By modifying the blind storage system in the EMRS, we solve the trapdoor unlinkability problem and conceal access pattern of the search user from the cloud server.
- We give thorough security analysis to demonstrate that the EMRS can reach a high security level including confidentiality of documents and index, trapdoor privacy, trapdoor unlinkability, and concealing access pattern of the search user. Moreover, we implement extensive experiments, which show that the EMRS can achieve enhanced efficiency in the terms of functionality and search efficiency compared with existing proposals.

The remainder of this paper is organized as follows. In Section II, the system model, security requirements and design goal are formalized. In Section III, we recap relevance scoring, secure kNN technique, blind storage system and ciphertext policy attribute-based encryption. In Section IV, we propose the EMRS. Its security analysis and performance evaluation are presented in Section V and Section VI, respectively. In Section VII, we present related work. Finally, we conclude this paper in Section VIII.

II. SYSTEM MODEL, SECURITY REQUIREMENTS AND DESIGN GOAL

A. SYSTEM MODEL

As shown in Fig. 1, the system model in the EMRS consists of three entities: data owner, search users and cloud server. The data owner keeps a large collection of documents D to be outsourced to a cloud server in an encrypted form C. In the system, the data owner sets a keyword dictionary W which contains d keywords. To enable search users to query over the encrypted documents, the data owner builds the encrypted index F. Both the encrypted documents C and encrypted index F are stored on the cloud server through blind storage system.

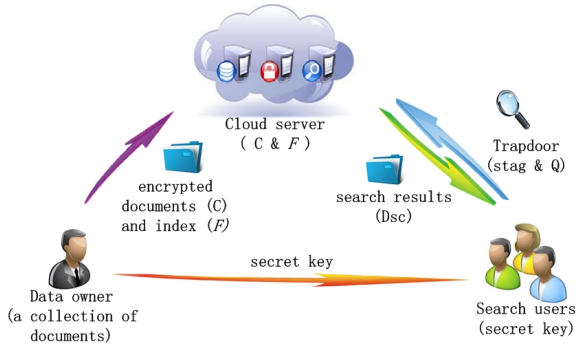


FIGURE 1. System model.

When a search user wants to search over the encrypted documents, she first receives the secret key from the data owner. Then, she chooses a conjunctive keyword set ϖ which contains l interested keywords and computes a trapdoor T including a keyword-related token *stag* and the encrypted query vector Q. Finally, the search user sends *stag*, Q, and an optional number k to the cloud server to request the most k relevant results.

Upon receiving *stag*, Q, and k from the search user, the cloud server uses the *stag* to access the index F in the blind storage and computes the relevance scores with the encrypted query vector Q. Then, the cloud server sends back descriptors (*Dsc*) of the top-k documents that are most relevant to the searched keywords. The search user can use these descriptors to access the blind storage system to retrieve the encrypted documents. An access control technique, e.g., attribute-based encryption, can be implemented to manage the search user's decryption capability.

B. SECURITY REQUIREMENTS

In the EMRS, we consider the cloud server to be curious but honest which means it executes the task assigned by the data owner and the search user correctly. However, it is curious about the data in its storage and the received trapdoors to obtain additional information. Moreover, we consider the *Knowing Background* model in the EMRS, which allows the cloud server to know more background information of the documents such as statistical information of the keywords.

Specifically, the EMRS aims to provide the following four security requirements:

- *Confidentiality of Documents and Index:* Documents and index should be encrypted before being outsourced to a cloud server. The cloud server should be prevented from prying into the outsourced documents and cannot deduce any associations between the documents and keywords using the index.
- *Trapdoor Privacy:* Since the search user would like to keep her searches from being exposed to the cloud server, the cloud server should be prevented from knowing the exact keywords contained in the trapdoor of the search user.
- *Trapdoor Unlinkability:* The trapdoors should not be linkable, which means the trapdoors should be totally different even if they contain the same keywords. In other words, the trapdoors should be randomized rather than determined. The cloud server cannot deduce any associations between two trapdoors.
- *Concealing Access Pattern of the Search User:* Access pattern is the sequence of the searched results. In the EMRS, the access pattern should be totally concealed from the cloud server. Specifically, the cloud server cannot learn the total number of the documents stored on it nor the size of the searched document even when the search user retrieves this document from the cloud server.

C. DESIGN GOAL

To enable efficient and privacy-preserving multi-keyword ranked search over encrypted mobile cloud data via blind storage system, the EMRS has following design goals:

- *Multi-Keyword Ranked Search:* To meet the requirements for practical uses and provide better user experience, the EMRS should not only support multi-keyword search over encrypted mobile cloud data, but also achieve relevance-based result ranking.
- *Search Efficiency:* Since the number of the total documents may be very large in a practical situation, the EMRS should achieve sublinear search with better search efficiency.
- *Confidentiality and Privacy Preservation:* To prevent the cloud server from learning any additional information about the documents and the index, and to keep search users' trapdoors secret, the EMRS should cover all the security requirements that we introduced above.

III. PRELIMINARIES

A. RELEVANCE SCORING

In searchable symmetric encryption (SSE) schemes, due to a large number of documents, search results should be retrieved in an order of the relevancy with the searched keywords. Scoring is the natural way to weight the relevancy of the documents. Among many relevance scoring techniques, we adopt *TF-IDF* weighting [15] in the EMRS. In *TF-IDF* weighting, term frequency $tf_{t,f}$ refers to the number of term t in a document f. Inverse document frequency is calculated as $idf_t = log\frac{N}{df_t}$, where df_t denotes the number of documents which contain term t and N refers to the total number of documents in the database. Then, the weighting of term t in a document f can be calculated as $tf_{t,f} * idf_t$.

B. SECURE kNN COMPUTATION

We adopt the work of Wong et al. [12] in the EMRS. Wong et al. propose a secure k-nearest neighbor (kNN) scheme which can confidentially encrypt two vectors and compute Euclidean distance of them. First, 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 [12].

C. BLIND STORAGE SYSTEM

A blind storage system [13] is built on the cloud server to support adding, updating and deleting documents and concealing the access pattern of the search user from the cloud server. In the blind storage system, all documents are divided into fixed-size blocks. These blocks are indexed by a sequence of random integers generated by a document-related seed. In the view of a cloud server, it can only see the blocks of encrypted documents uploaded and downloaded. Thus, the blind storage system leaks little information to the cloud server. Specifically, the cloud server does not know which blocks are of the same document, even the total number of the documents and the size of each document. Moreover, all the documents and index can be stored in the blind storage system to achieve a searchable encryption scheme.

D. CIPHERTEXT POLICY ATTRIBUTE-BASED ENCRYPTION

In ciphertext policy attribute-based encryption (CP-ABE) [16], ciphertexts are created with an access structure (usually an access tree) which defines the access policy. A user can decrypt the data only if the attributes embedded in his attribute keys satisfy the access policy in the ciphertext. In CP-ABE, the encrypter holds the ultimate authority of the access policy.

IV. PROPOSED SCHEME

In this section, we propose the detailed EMRS. Since the encrypted documents and index $\vec{I}$ are both stored in the blind storage system, we would provide the general construction of the blind storage system. Moreover, since the EMRS aims to eliminate the risk of sharing the key that is used to encrypt the documents with all search users and solve the trapdoor unlinkability problem in Naveed's scheme [13], we modify the construction of blind storage and leverage ciphertext policy attribute-based encryption (CP-ABE) technique in the EMRS. However, specific construction of CP-ABE is out of scope of this paper and we only give a simple indication here. The notations of this paper is

TABLE 1. Notations.

Symbols	Meanings
D	collection of m documents
C	collection of m encrypted documents
W	keyword dictionary containing d keywords
p, P	relevance vector and its encrypted form
q, Q	query vector and its encrypted form
ϖ	a conjunctive keyword set for search request
$stag$	a keyword-related token
B	an array of n_b blocks of m_b bits each
F	an efficient search index
$Enc_K()$	symmetric encryption algorithm using key K

shown in Table 1. The EMRS consists of the following phases: System Setup, Construction of Blind Storage, Encrypted Database Setup, Trapdoor Generation, Efficient and Secure Search, and Retrieve Documents from Blind Storage.

A. SYSTEM SETUP

The data owner takes a security parameter λ, and outputs two invertible matrixes $M_1, M_2 \in R^{(d+2)*(d+2)}$ as well as a (d+2)-dimension binary vector S as the secret key, where d represents the size of the keyword dictionary. Then, the data owner generates a set of attribute keys sk for each search user according to her role in the system. The data owner chooses a key K_T for a symmetric cryptography $Enc()$, e.g., AES. Finally, the data owner sends $(M_1, M_2, S, sk, Enc(), K_T)$ to the search user through a secure channel.

B. CONSTRUCTION OF BLIND STORAGE

The data owner chooses a full-domain collusion resistant hash function H, a full-domain pseudorandom function Ψ, a pseudorandom generator Γ and a hash function $\Phi: \{0,1\}^* \rightarrow \{0,1\}^{192}$. Ψ and Γ are based on the AES block-cipher [13]. Then, the data owner chooses a number $\alpha > 1$ that defines the expansion parameter and a number κ that denotes the minimum number of blocks in a communication.

1) B.KEYGEN

The data owner generates a key K_Ψ for the function Ψ and sends it to the search user using a secure channel.

2) B.BUILD

This phase takes into a large collection of documents D. D is a list of documents $(d_1, d_2, d_3 \cdots d_m)$ containing m documents. where each document has a unique id denoted as id_i. The B.Build outputs an array of blocks B, which consists of n_b blocks of m_b bits each. For document d_i, it contains $size_i$ blocks of m_b bits each and each header of these blocks contains the $H(id_i)$. In addition, the header of the first block of the document d_i indicates the size of d_i. At the beginning, we initialize all blocks in B with all 0. For each document d_i in D, we construct the blind storage as follows:

Step 1: Compute the seed $\sigma_i = \Psi_{K_\Psi}(id_i)$ as the input of the function Γ. Generate a sufficiently long bit-number through the function Γ using the seed σ_i and parse it as a sequence of integers in the range $[n_b]$. Let $\pi[\sigma_i, l]$ denote the first l integers of this sequence. Generate a set $S_f = \pi[\sigma_i, \max(\lceil \alpha * size_i \rceil, \kappa)]$.

Step 2: Let $S_f^0 = \pi[\sigma_i, \kappa]$, then check if the following conditions hold:

- There exists $size_i$ free blocks indexed by the integers in the set S_f.
- There exists one free block indexed by the integers in the set S_f^0.

If either of the above two does not hold, abort.

Step 3: Pick a subset $S_f' \subset S_f$ that contains $size_i$ integers, and make sure that the blocks indexed by these integers in the subset S_f' are all free. We would rely on the fact that integers in the set S_f are in a random order and we pick the first $size_i$ integers indexing free blocks and make these integers form the subset S_f'. Mark these blocks as unfree. Then, write the document d_i to the blocks indexed by the integers in S_f' in an increasing order.

Note that, one can once write the blocks of different documents to the blind storage system to conceal the associations of the blocks. Moreover, the specific construction of each block and the encryption of the blocks would be discussed next.

DISCUSSIONS

The main idea of the blind storage system is that storing a document in a set of fixed-size blocks indexed by the integers, that are generated by applying the seed σ_i to the pseudorandom generator Γ. To reduce the probability that the number of free blocks indexed by integers in S_f is less than $size_i$, we can choose a sequence of $\alpha * size_i$ integers as the set S_f. Here the choice of the parameter α is an inherent tension between collision probability and the wasted space. And the probability the above two conditions in Step 2 do not hold may be negligible by the choice of the parameters [13]. And we would prove it in Section V.

C. ENCRYPTED DATABASE SETUP

The data owner builds the encrypted database as follows:

Step 1: The data owner computes the d-dimension relevance vector $p = (p_1, p_2, \cdots p_d)$ for each document using the *TF-IDF* weighting technique, where p_j for $j \in (1, 2 \cdots d)$ represents the weighting of keyword ω_j in document d_i. Then, the data owner extends the p to a (d+2)-dimension vector p^*. The (d+1)-th entry of p^* is set to a random number ε and the (d+2)-th entry is set to 1. We would let ε follow a normal distribution $N(\mu, \sigma^2)$ [11]. For each document d_i, to compute the encrypted relevance vector, the data owner encrypts the associated extended relevance vector p^* using the secret key M_1, M_2 and S. First, the data owner chooses a random number r and splits the extended relevance vector p^* into two (d+2)-dimension vectors p' and p'' using the vector S. For the j-th item in p^*, set

$$\begin{cases} p_j' = p_j'' = p_j^*, & if\ S_j = 1 \\ p_j' = \frac{1}{2}p_j^* + r, & p_j'' = \frac{1}{2}p_j^* - r, \quad otherwise \end{cases} \tag{1}$$

where S_j represents the j-th item of S. Then compute the $P = \{M_1^T \cdot p', M_2^T \cdot p''\}$ as the encrypted relevance vector.

Step 2: For each document d_i in D, set the document into blocks of m_b bits each. For each block, there is a header $H(id_i)$ indicating that this block belongs to document d_i. And the $size_i$ of the document is contained in the header of the first block of d_i. Then, for each document d_i, the data owner chooses a 192-bit key K_i for the algorithm $Enc()$. More precisely, for each block $B[j]$ of the document d_i, where j represents the index number of this block, compute the $K_i \oplus \Phi(j)$ as the key for the encryption of this block. Since each block has a unique index number, the blocks of the same document are encrypted with different keys. The document d_i contains $size_i$ encrypted blocks and the first block of the document d_i with index number j is as

$$Enc_{(K_i \oplus \Phi(j))}(H(id_i)||size_i||data) \tag{2}$$

And the rest of the blocks of d_i is as

$$Enc_{(K_i \oplus \Phi(j))}(H(id_i)||data) \tag{3}$$

Finally, the data owner encrypts all the documents and writes them to the blind storage system using the B.Build function.

Step 3: To enable efficient search over the encrypted documents, the data owner builds the index $\digamma$. First, the data owner defines the access policy υ_i for each document d_i. We denote the result of attribute-based encryption using access policy υ_i as $ABE_{\upsilon_i}()$. The data owner initializes $\digamma$ to an empty array indexed by all keywords. Then, the index $\digamma$ can be constructed as shown in Algorithm 1.

Algorithm 1 Initialize $\digamma$

```
for each keyword ω ∈ W do
    Set t an empty list
    for each document d_i containing the keyword ω do
        Get the associated vector P of d_i
        Choose a random number x
        Dsc ← ABE_{υ_i}(id_i||K_i||x)
        Append the tuple (Dsc, P) to t
    end for
    F[ω] = t
end for
return F
```

As we can see, the index $\digamma$ maps the keyword to the encrypted relevance vectors (P) and the descriptors (Dsc) of the documents that contain the keyword. And each list $\digamma[\omega]$ can be transformed to be stored in the blind storage system with ω as the document id. Specifically, for each $\digamma[\omega]$, the data owner computes $\sigma_\omega = \Psi_{K_\Psi}(\omega)$ as the seed for the function Γ to generate the set S_f. Here, for each block of $\digamma[\omega]$ indexed by the integer j, the data owner adds an encrypted header as $Enc_{(K_T \oplus \Phi(j))}(H(\omega)||size_\omega)$, where $size_\omega$ represents the number of blocks that belong to $\digamma[\omega]$. Finally, the data owner writes the index $\digamma$ to the blind storage system using the B.Build function.

DISCUSSIONS

When using the B.Build function, it is crucial to determine the way we compute the seed for generating the set S_f. We use the document id id_i to compute the seed for the documents stored in the blind storage system, and the keyword ω to compute the seed for each $\digamma[\omega]$. Moreover, each header of the blocks of the documents contains the encrypted $H(id_i)$ and the first block indicates the $size_i$. And the blocks of index $\digamma$ are different from those of the documents. Each header of the blocks of index $\digamma$ is denoted as $Enc_{(K_T \oplus \Phi(j))}(H(\omega)||size_\omega)$. This little change is for the security concerns and does not affect the implementation of the blind storage. In addition, since each block is encrypted using the key generated by the index number, the headers would be different even if the two blocks belong to the same document or the same list $\digamma[\omega]$.

D. TRAPDOOR GENERATION

To search over the outsourced encrypted data, the search user needs to compute the trapdoor including a keyword-related token *stag* and encrypted query vector Q as follows:

Step 1: The search user takes a keyword conjunction $\varpi = (\omega_1, \omega_2, \cdots \omega_l)$ with l keywords of interest in W. A d-dimension binary query vector q is generated where the j-th bit of q represents whether $\omega_j \in \varpi$ or not. Then, the search user chooses two random numbers r, t and scales the query vector q to a (d+2)-dimension vector q^* as

$$q^* = \{rq, r, t\} \tag{4}$$

Then, the search user chooses a random number r' and splits the vector q^* into two (d+2)-dimension vectors q' and q''. For the j-th item in q^*, set

$$\begin{cases} q'_j = q''_j = q^*_j, & if\ S_j = 0 \\ q'_j = \frac{1}{2}q^*_j + r', & q''_j = \frac{1}{2}q^*_j - r', \quad otherwise \end{cases} \tag{5}$$

The search user computes the $Q = \{M_1^{-1} \cdot q', M_2^{-1} \cdot q''\}$ as the encrypted query vector.

Step 2: The search user chooses the estimated least frequent keyword ω' in the conjunction ϖ and computes the seed $\sigma_{\omega'} = \Psi_{K_\Psi}(\omega')$. Then the search user generates a long bit-number through the function Γ using the seed $\sigma_{\omega'}$. The search user chooses the sequence $\pi[\sigma_{\omega'}, \kappa]$ and randomly adds κ dummy integers to the sequence. The search user downloads the blocks indexed by these 2κ integers and decrypts the header using the key $K_T \oplus \Phi(j)$, where j is the index number of the block, to find the first block of the list $\digamma[\omega']$, which consists of the descriptors and the encrypted relevance vectors of the documents containing ω'. Then the search user obtains the $size_{\omega'}$ from the first block and computes the set $S_\omega = \pi[\sigma_{\omega'}, \alpha * size_{\omega'}]$. The search user randomly adds $\alpha * size_{\omega'}$ dummy integers to the set S_ω resulting in a set S'_ω of $2\alpha * size_{\omega'}$ integers. And the extended set S'_ω is denoted as *stag*. Note that, the *stag* consists of some dummy integers, which is for the privacy consideration.

Finally, the search user sends Q, *stag* and a number k to the cloud server to request the most k relevant documents.

E. EFFICIENT AND SECURE SEARCH

Upon receiving Q, *stag*, and k, the cloud server parses the *stag* to get a set of integers in the range $[n_b]$. Then, the cloud server accesses index F in the blind storage and retrieves the blocks indexed by the integers to obtain the tuples $(ABE_{\upsilon_i}(id_i||K_i||x), P)$ on these blocks. Note that, these blocks consist of the blocks of $F[\omega']$ and some dummy blocks. For each retrieved encrypted relevance vector P, compute the relevance score $Score_i$ for the associated document d_i with the encrypted query vector Q as follows:

$$\begin{aligned} Score_i &= P \cdot Q \\ &= \{M_1^T \cdot p', M_2^T \cdot p''\} \cdot \{M_1^{-1} \cdot q', M_2^{-1} \cdot q''\} \\ &= p' \cdot q' + p'' \cdot q'' \\ &= p^* \cdot q^* \\ &= (p, \epsilon, 1) \cdot (rq, r, t) \\ &= r(pq + \varepsilon) + t \end{aligned} \tag{6}$$

Finally, after sorting the relevance scores, the cloud server sends back the descriptors $ABE_{\upsilon_i}(id_i||K_i||x)$ of the top-k documents that are most relevant to the searched keywords. Note that, as discussed before, attribute-based encryption as an access control technique can be implemented to manage search user's decryption capability.

F. RETRIEVE DOCUMENTS FROM BLIND STORAGE

Upon receiving a set of descriptors $ABE_{\upsilon_i}(id_i||K_i||x)$, the search user can retrieve the documents as follows:

Step 1: If the search user's attributes satisfy the access policy of the document, the search user can decrypt the descriptor using her secret attribute keys to get the document id id_i and the associated symmetric key K_i. To retrieve the document d_i, compute $\sigma_i = \Psi_{K_\Psi}(id_i)$ for the function Γ. Generate a sufficiently long bit-number through the function Γ using the seed σ_i, parse it as a sequence of integers in the range $[n_b]$ and choose the first κ integers as the set S_f^0. Retrieve the blocks indexed by these κ integers from the encrypted database D through blind storage system.

Step 2: The search user tries to decrypt these blocks using the symmetric key $K_i \oplus \Phi(j)$, until she finds the first block of the document d_i. If she does not find the first block, the document is not accessed in the system. Otherwise, the search user recovers the size of the document $size_i$ from the header of the first block.

Step 3: Then, the search user computes $l = \lceil \alpha * size_i \rceil$. If $l \leq \kappa$, compute $S_f = \pi[\sigma_i, \kappa]$. Otherwise, compute $S_f = \pi[\sigma_i, l]$ and retrieve the rest of the blocks indexed by the integers in S_f via the blind storage system. Decrypt these blocks and combine the blocks with the header $H(id_i)$ in an increasing order to recover document d_i.

DISCUSSIONS

Here we explain how the search user retrieves one document from the blind storage system. This can form the foundation of the B.Aceess function of the blind storage. Moreover, the search user can require more than one document once by combining the sequence S_f^0 and S_f of different documents in a random order. And this combination can further conceal access pattern of the search user since the cloud server even does not know the number of documents that the search user requires.

V. SECURITY ANALYSIS

Under the assumption presented in Section II, we analyze the security properties of the EMRS. We give analysis of the EMRS in terms of confidentiality of documents and index, trapdoor privacy, trapdoor unlinkability and concealing access pattern of the search user.

A. CONFIDENTIALITY OF DOCUMENTS AND INDEX

The documents are encrypted by the traditional symmetric cryptography technique before being outsourced to the cloud server. Without a correct key, the search user and cloud server cannot decrypt the documents. As for index confidentiality, the relevance vector for each document is encrypted using the secret key M_1, M_2, and S. And the descriptors of the documents are encrypted using CP-ABE technique. Thus, the cloud server can only use the index F to retrieve the encrypted relevance vectors without knowing any additional information, such as the associations between the documents and the keywords. And only the search user with correct attribute keys can decrypt the descriptor $ABE_{\upsilon_i}(id_i||K_i||x)$ to get the document id and the associated symmetric key. Thus, the confidentiality of documents and index can be well protected.

B. TRAPDOOR PRIVACY

When a search user generates her trapdoor including the keyword-related token *stag* and encrypted query vector Q, she randomly chooses two numbers r and t. Then, for the query vector q, the search user extends it as (rq, r, t) and encrypts the query vector using the secret key M_1, M_2 and S. Thus, the query vectors can be totally different even if they contain same keywords. And we use the secure function Ψ and Γ to help the search user compute keyword-related token *stag* using the secret key K_Ψ. Without the secret key M_1, M_2, S and K_Ψ, the cloud server cannot pry into the trapdoor. And the search user can add dummy integers to the set S_f to conceal what she is truly searching for. Thus, the keyword information in the trapdoor is totally concealed from the cloud server in the EMRS and trapdoor privacy is well protected.

C. TRAPDOOR UNLINKABILITY

Trapdoor unlinkability is defined as that the cloud server cannot deduce associations between any two trapdoors. Even though the cloud server cannot decrypt the trapdoors, any association between two trapdoors may lead to the leakage of the search user's privacy. We consider whether the two trapdoors including *stag* and the encrypted query vector Q can be linked to each other or to the keywords. Moreover, we

would prove the EMRS can achieve trapdoor unlinkability under the *Knowing Backgroud* model.

To compute the encrypted query vector Q that is defined as $\{M_1^{-1} \cdot q', M_2^{-1} \cdot q''\}$ in the EMRS. First, the search user needs to extend the query vector q to q^*. As we can see, the (d+1)-th and (d+2)-th entry of the vector q^* are set to random values r and t. So there are $2^{\eta_r} * 2^{\eta_t}$ possible values, where the number r and t are η_r-bit or η_t-bit long, respectively. Further, the search user needs to split the vector q^* according to the splitting vector S as we discussed above. If $S_j = 0$, the q_j^* is split into two random values which add up to q_j^*. Suppose that the number of 0 in S is μ and each dimension of the vector q' is η_q-bit long. We can see that η_r, η_t, μ and η_q are independent of each other. Then we can compute the probability that two encrypted query vectors are the same as

$$P = \frac{1}{2^{\eta_r} 2^{\eta_t} 2^{\mu\eta_q}} = \frac{1}{2^{\eta_r+\eta_t+\mu\eta_q}} \tag{7}$$

Therefore, the larger these parameters are, the lower the probability is. Hence, if we choose 1024-bit r and t, the probability that two encrypted query vectors are the same is $P < \frac{1}{2^{2048}}$, which is negligible as a result.

As for the keyword-related token *stag*, the search user first obtains the $size_\omega$ from the cloud server using the sequence of 2κ integers, half of which are dummy integers. Then, the search user computes the set $S_\omega = \pi[\sigma_\omega, \alpha * size_\omega]$ and adds $\alpha * size_\omega$ dummy integers to the set S_ω to form the *stag*. Thus, each *stag* contains $2\alpha * size_\omega$ random integers, half of which are random integers. Suppose the integers are n_b bits long. Then the probability that the two *stag*s are the same is

$$P' = \frac{1}{2^{2\alpha * size_\omega * n_b}} \tag{8}$$

Hence, if we choose 12-bit long n_b, 3-bit long extension parameter α and $size_\omega$ is supposed to be 8-bit long, the probability $P' < \frac{1}{2^{576}}$, which is negligible as a result.

In Cash's scheme [10] and Naveed's scheme [13], for the same keyword, the search user can only compute the same *stag* or the same set S_f. Moreover, when a search user accesses the cloud server using a keyword that has been searched before, the cloud server can learn that two search requests contain the same keyword. Under *Knowing Backgroud* model, the cloud server may learn the search frequency of the keywords and deduce some information using the statistic knowledge in [10] and [13].

D. CONCEALING ACCESS PATTERN OF THE SEARCH USER

The access pattern means the sequence of the searched results [11]. In Cash's scheme [10] and Cao's scheme [11], the search user directly obtains the associated documents from the cloud server, which may reveal the association between the search request and the documents to the cloud server. In the EMRS by modifying the blind storage system, access pattern is well concealed from the cloud server. Since the headers of the blocks are encrypted with the block number j and each descriptor has a random padding, they would be different even if they belong to the same document. Thus, in view of the cloud server, it can only see blocks downloaded and uploaded. And, the cloud server even does not know the number of the documents stored in its storage and the length of each document, since all the documents are divided into blocks in a random order. In addition, when a search user requests a document, she can choose more blocks than the document contains. Moreover, she can require blocks of different documents at one time in a random order to totally conceal what she is requesting.

In the implementation of the blind storage system, there would be a trade-off between security guarantee and performance by the choice of parameters. We define the P_{err} as the probability that the data owner aborts the document when there are not enough free blocks indexed by the integers in the set S_f as discussed in Section IV. When this abort happens, some illegitimate information may be revealed to the cloud server [13]. We consider the following parameters γ, α and κ to measure the P_{err}. We denote $\gamma = n_b/m$, where n_b is the number of blocks in the array B and m is the total number of the documents stored on the cloud server. α is the ratio that scales the number of blocks a document contains to the number of blocks in the set S_f. κ is the minimum number of blocks in a transaction. Then, according to [13], we can compute the P_{err} as

$$P_{err}(\gamma, \alpha, \kappa) \leq \max_{n \geq \frac{\kappa}{\alpha}} \sum_{i=0}^{n-1} \binom{\lceil \alpha n \rceil}{i} \left(\frac{\gamma - 1}{\gamma}\right)^i \left(\frac{1}{\gamma}\right)^{\lceil \alpha n \rceil - i} \tag{9}$$

As we can see, the higher these parameters we choose, the lower the probability P_{err} is and the higher the security guarantee would be. However, the parameters also influence the performance of the blind storage system, such as the communication and computation cost. By the choice of these parameters, the probability P_{err} would be negligible [13].

The comparison of security level is shown in TABLE 2. We can see that the EMRS can achieve best security guarantees compared with the exiting schemes [10], [11], [13].

TABLE 2. Comparison of security level.

	[10]	[11]	[13]	EMRS
Confidentiality	√	√	√	√
Trapdoor Unlinkability		√		√
Concealing Access Pattern of the Search User			√	√

VI. PERFORMANCE EVALUATION

A. FUNCTIONALITY

Considering a large number of documents and search users in a cloud environment, searchable encryption schemes should allow privacy-preserving multi-keyword search and return documents in a order of higher relevance to the search request. As shown in TABLE 3, we compare functionalities among

the EMRS, Cash's scheme [10], Cao's scheme [11] and Naveed's scheme [13].

TABLE 3. Comparison of functionalities.

	[10]	[11]	[13]	EMRS
Multi-keyword	√	√		√
Result Ranking		√		√
Relevance Scoring		√		√

Cash's scheme supports multi-keyword search, but cannot return results in a specific order of the relevance score. Cao's scheme achieves multi-keyword search and returns documents in a relevance-based order. Naveed's scheme implements the blind storage system to protect the access pattern but it only supports single-keyword search and returns undifferentiated results. The EMRS can achieve multi-keyword search, and relevance sorting while preserving a high security guarantees as discussed in Section V.

B. COMPUTATION OVERHEAD

We evaluate the performance of the EMRS through simulations and compare the time cost with Cao's [11]. We apply a real dataset National Science Foundation Research Awards Abstracts 1990-2003 [17], by randomly selecting some documents. Then, we conduct real-world experiments on a 2.8Hz-processor, computing machine to evaluate the performance of index construction and search phases. Moreover, we implement the trapdoor generation on a 1.2GHz smart phone. We would show the simulation experiments of the EMRS, and demonstrate that the computation overhead of index construction and trapdoor generation are almost the same compared with that of Cao's [11]. Then we would compare the execution time of search phase with Cao's [11] and show that the EMRS achieves better search efficiency.

1) INDEX CONSTRUCTION

Index construction in the EMRS consists of two phases: encrypted relevance vector computation and the efficient index $\mathcal{F}$ construction via blind storage.

As for the computation of encrypted relevance vector, the data owner first needs to compute the relevance score for each keyword in each document using the $TF - IDF$ technique. As shown in Fig. 2, both the size of the dictionary and the number of documents would influence the time for calculating all the relevance scores. Then, to compute the encrypted relevance vector P, the data owner needs two multiplications of a $(d+2)*(d+2)$ matrix and a (d+2)-dimension vector with complexity $O(d^2)$. The time cost for computing all the encrypted relevance vectors is linear to the size of the database since time for building subindex of one document is fixed. Thus, the computation complexity is $O(md^2)$, where m represents the number of documents in the database and d represents the size of the keyword dictionary W. The computation complexity is as the same as that in Cao's [11]. The computational cost for computing the encrypted relevance vectors is shown in Fig. 3. As we can see, both the size of the dictionary and the number of documents would affect the execution time.

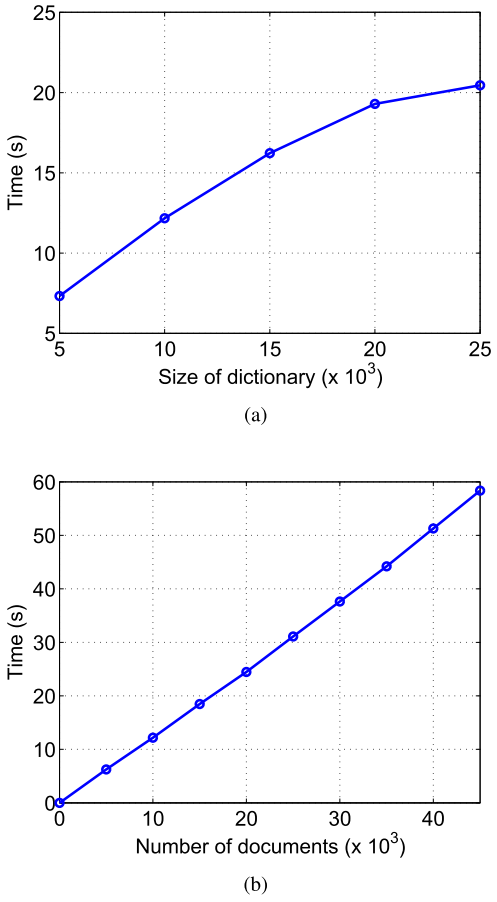


FIGURE 2. Time for calculating relevance score. (a) For the different size of dictionary with the same number of documents, $m = 10000$. (b) For the different number of documents with the same size of dictionary, $|W| = 10000$.

Finally, we adopt the index $\mathcal{F}$ via the blind storage in the EMRS to improve search efficiency and conceal the access pattern of the search user. For each keyword $\omega \in W$, we need to build the list $\mathcal{F}[\omega]$ of tuples $(ABE_{v_i}(id_i||K_i||x), P)$ of documents that contain the keyword and upload it using the B.Build function. So the computation complexity to build the index $\mathcal{F}$ is $O(\varrho d)$, where ϱ represents the average number of tuples contained in the list $\mathcal{F}[\omega]$ and is no more than the number of document m. Since the access pattern is not considered in most schemes, we are not going to give the specific comparison of the implementation of the blind storage [13] in the EMRS.

2) TRAPDOOR GENERATION

In the EMRS, trapdoor generation consists of *stag* and encrypted query vector Q. To compute *stag*, the search user only needs two efficient operations (Ψ and Γ) to generate a sequence of random integers. Compared with time cost to compute the encrypted query vector which is linearly

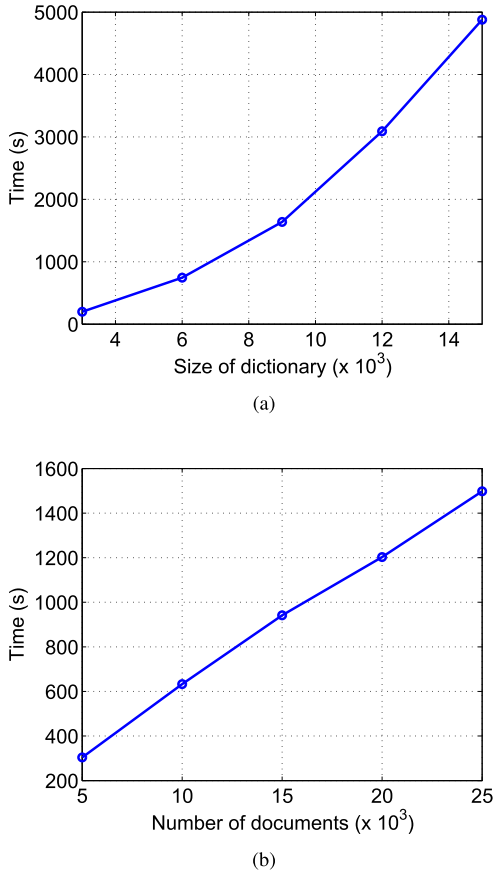


FIGURE 3. **Time for computing the encrypted relevance vectors. (a) For the different size of dictionary with the same number of documents, $m = 6000$. (b) For the different number of documents with the same size of dictionary, $|W| = 4000$.**

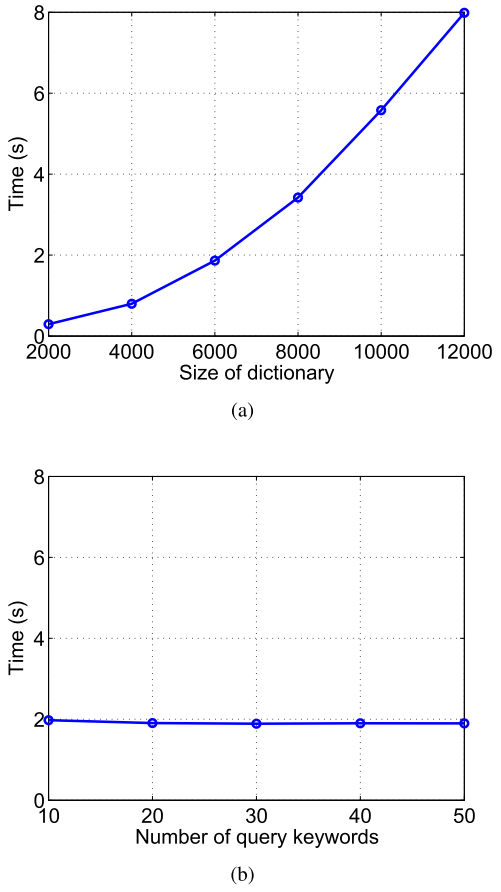


FIGURE 4. **Time for generating trapdoor on a real smart phone. (a) For the different size of dictionary with the same number of query keywords, $|\varpi| = 20$. (b) For the different number of query keywords with the same size of dictionary, $|W| = 6000$.**

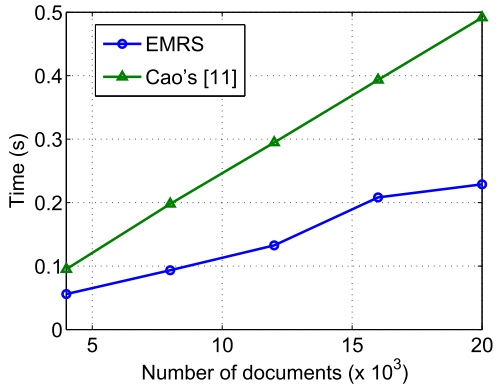
increasing with the size of the keyword dictionary, time cost for computing *stag* is negligible. As for computing the encrypted query vector Q, the search user needs to compute two multiplications of a $(d + 2) * (d + 2)$ matrix and a (d+2)-dimension vector with complexity $O(d^2)$. Thus, the computation complexity of trapdoor generation for the search user is $O(d^2)$, which is as the same as that in Cao's scheme [11]. As shown in Fig. 4, we conduct a simulation experiment on a 1.2Ghz smart phone and give the experiment results for computing trapdoor in the EMRS.

3) SEARCH EFFICIENCY

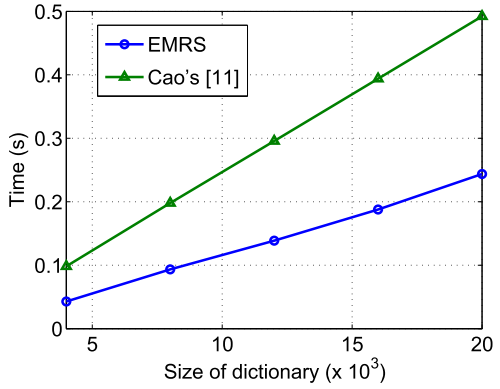
Search operation in Cao's scheme [11] requires computing the relevance scores for all documents in the database. For each document, the cloud server needs to compute the inner product of two (d+2)-dimension vectors twice. Thus, the computation complexity for the whole data collection is $O(md)$. As we can see, the search time in Cao's scheme linearly increases with the scale of the dataset, which is impractical for large-scale dataset.

In the EMRS, by adopting the inverted index I which is built in the blind storage system, we achieve a sublinear computation overhead compared with Cao's scheme. Upon receiving *stag*, the cloud server can use *stag* to access blind storage and retrieve the encrypted relevance vector on the blocks indexed by the *stag*. These blocks consist of blocks of documents containing the *stag*-related keyword and some dummy blocks. Thus, the EMRS can significantly decrease the number of documents which are relevant to the searched keywords. Then, the cloud server only needs to compute the inner product of two (d+2)-dimension vectors for the associated documents rather than computing relevance scores for all documents as that in Cao's scheme [11]. The computation complexity for search operation in the EMRS is $O(\alpha\varrho_s d)$, where ϱ_s represents the the number of documents which contain the keyword applied by the keyword-related token *stag* and the α is the extension parameter that scales the number of blocks in a document to the number of blocks in the set S_f. The value of ϱ_s can be small if the search user typically chooses the estimated least frequent keyword, such that the computation cost for search on the cloud server is significantly reduced.

As shown in Fig. 5, the computation cost of search phase is mainly affected by the number of documents in the dataset and the size of the keyword dictionary. In our experiments, we implement the index on the memory to avoid the time-cost



(a)



(b)

FIGURE 5. Time for search on the cloud server. (a) For different number of documents with the same size of keyword dictionary and number of searched keywords, $|W| = 8000$, $|\varpi| = 20$. (b) For different size of keyword dictionary with the same number of documents and searched keywords, $m = 8000$, $|\varpi| = 20$.

I/O operations. Note that, although the time costs of search operation are linearly increasing in both schemes, the increase rate of the EMRS is less than half of that in Cao's scheme.

C. COMMUNICATION OVERHEAD

When the system is once setup, including generating encrypted documents and index, the communication overhead is mainly influenced by the search phase. In this section, we would compare the communication overhead among the EMRS, Cash's scheme [10], Cao's scheme [11] and Naveed's scheme [13] when searching over the cloud server. Since most existing schemes of SSE only consider obtaining a sequence of results rather than the related documents, the comparison here would not involve the communication of retrieving the documents.

In Cao's scheme [11], the search user needs to compute the trapdoor and send it to the cloud server. Then it can obtain the searched results. The communication overhead in Cao's is $2(d+2)\eta_q$, where d represents the size of the keyword dictionary and each dimension of the encrypted query vector is η_q-bit long. According to Cash's scheme [10], when a search user wants to query over the cloud server using a conjunctive keyword set ϖ, she needs to compute *stag* for the estimated least-frequent keyword and *xtoken*s for the other keywords in the set ϖ. And, each *xtoken* contains $|\varpi|$ elements in G, where G is a group of prime order p. Moreover, the search user needs to continuously compute the *xtoken* until the cloud server sends stop, which indicates that the total number of the *xtoken*s is linear to ϱ, the number of documents containing the keyword related to the *stag*. This results in much unnecessary communication overhead of $\varrho|\varpi||G|$, where $|G|$ represents the size of an element in G. In Naveed's scheme [13], since the index is constructed in the blind storage system, the search user may need to access the blind storage system to obtain the $size_\omega$ and then obtain the results. This requires one or two round communication of $\alpha * size_\omega * n_b$ bits, where α is the extension parameter, $size_\omega$ is the number of blocks of documents containing ω, and each index number is n_b-bit long. In the EMRS, we modify the way the search user computes the sequence S_f that indexes the blocks by adding some dummy integers to S_f to conceal what the search user is searching for. The communication comparison is shown in TABLE 4. As we can see, even though the EMRS requires a little more communication overhead, the EMRS can achieve more functionalities compared with [10], [13] as shown in TABLE 3 and better search efficiency compared with [11] as shown in Fig. 5.

TABLE 4. Comparison of communication overhead.

	Number of Rounds	Size of Message(bit)
Cash's [10]	many	$\varrho\|\varpi\|\|G\|$
Cao's [11]	one	$2(d+2)\eta_q$
Naveed's [13]	one/two	$\alpha * size_\omega * n_b$
EMRS	one/two	$2(d+2)\eta_q + 2\alpha * size_\omega * n_b$

DISCUSSIONS

Note that the communication overhead in our paper is higher than that in the Cao's scheme. But the higher communication overhead will not severely affect the user's experience. This is because that the communication overhead is mainly incurred by the exchange of short signaling messages and can be transmitted in a very short time. Moreover, with the adoption of advanced wireless technology, such as 4G/5G and IEEE 802.11ac, the communication delays tend to further reduce and negligible. As a theoretical framework, in this paper, we target to a prototype system and expose our proposal to the public. As such, based on the specific deployment scenarios, e.g., whether communication bandwidth is expensive and precious or not, to modify our proposal for real-world implementation.

D. SIZE OF RETURNED RESULTS

The size of the returned results in the EMRS is mainly affected by the choice of the security parameters, α and κ. The larger these two numbers are, the higher security guarantee the scheme provides, as we discussed in Section V.

The size of returned results for each document can be $a*size_\omega$ blocks, which contain the blocks of searched document and dummy blocks. Moreover, the search user can require many documents at one time and thus can avoid requesting dummy blocks. The EMRS provides balance parameters for search users to satisfy their different requirements on communication and computation cost, as well as privacy.

VII. RELATED WORK

Searchable encryption is a promising technique that provides the search service over the encrypted cloud data. It can mainly be classified into two types: Searchable Public-key Encryption (SPE) and Searchable Symmetric Encryption (SSE).

Boneh et al. [18] first propose the concept of SPE, which supports single-keyword search over the encrypted cloud data. The work is later extended in [19] to support the conjunctive, subset, and range search queries on encrypted data. Zhang et al. [20] propose an efficient public key searchable encryption scheme with conjunctive-subset search. However, the above proposals require that the search results match all the keywords at the same time, and cannot return results in a specific order. Further, Liu et al. [21] propose a ranked search scheme which adopts a mask matrix to achieve cost-effectiveness. Yu et al. [15] propose a multi-keyword retrieval scheme that can return the top-k relevant documents by leveraging the fully homomorphic encryption. [22], [23] adopt the attribute-based encryption technique to achieve search authority in SPE.

Although SPE can achieve above rich search functionalities, SPE are not efficient since SPE involves a good many asymmetric cryptography operations. This motivates the research on SSE mechanisms.

The first SSE scheme is introduced by Song et al. [24], which builds the searchable encrypted index in a symmetric way but only supports single keyword. Curtmola et al. further improve the security definitions of SSE in [25]. Their work forms the basis of many subsequent works, such as [10], [13], and [26], by introducing the fundamental approach of using a keyword-related index, which enable the quickly search of documents that contain a given keyword. To meet the requirements of practical uses, conjunctive multi-keyword search is necessary which has been studied in [11] and [15]. Moreover, to give the search user a better search experience, some proposals [27], [28] propose to enabled ranked results instead of returning undifferentiated results, by introducing the relevance score to the searchable encryption. To further improve the user experience, fuzzy keyword search over the encrypted data has also been developed in [7] and [29].

Cao et al. [11] propose a privacy-preserving multi-keyword search scheme that supports ranked results by adopting secure k-nearest neighbors (kNN) technique in searchable encryption. The proposal can achieve rich functionalities such as multi-keyword and ranked results, but requires the computation of relevance scores for all documents contained in the database. This operation incurs huge computation overload to the cloud server and is therefore not suitable for large-scale datasets. Cash et al. [10] adopt the inverted index $TSet$, which maps the keyword to the documents containing it, to achieve efficient multi-keyword search for large-scale datasets. The works is later extended in [26] with the implementation on real-world datasets. However, the ranked results is not supported in [26]. Naveed et.al. [13] construct a blind storage system to achieve searchable encryption and conceal the access pattern of the search user. However, only single-keyword search is supported in [13].

VIII. CONCLUSION

In this paper, we have proposed a multi-keyword ranked search scheme to enable accurate, efficient and secure search over encrypted mobile cloud data. Security analysis have demonstrated that proposed scheme can effectively achieve confidentiality of documents and index, trapdoor privacy, trapdoor unlinkability, and concealing access pattern of the search user. Extensive performance evaluations have shown that the proposed scheme can achieve better efficiency in terms of the functionality and computation overhead compared with existing ones. For the future work, we will investigate on the authentication and access control issues in searchable encryption technique.

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 Trans. Veh. Technol.*, vol. 61, no. 5, pp. 2222–2232, Jun. 2012.

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

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

[4] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[5] H. Li, Y. Dai, L. Tian, and H. Yang, "Identity-based authentication for cloud computing," in *Cloud Computing*. Berlin, Germany: Springer-Verlag, 2009, pp. 157–166.

[6] W. Sun, *et al.*, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, 2013, pp. 71–82.

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

[8] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in *Proc. NDSS*, Feb. 2014.

[9] Y. Yang, H. Li, W. Liu, H. Yang, and M. Wen, "Secure dynamic searchable symmetric encryption with constant document update cost," in *Proc. GLOBECOM*, Anaheim, CA, USA, 2014.

[10] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Proc. CRYPTO*, 2013, pp. 353–373.

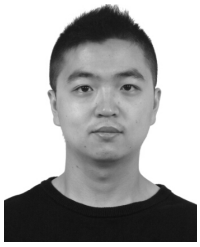
[11] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.

[12] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 139–152.

[13] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 639–654.

[14] H. Pang, J. Shen, and R. Krishnan, "Privacy-preserving similarity-based text retrieval," *ACM Trans. Internet Technol.*, vol. 10, no. 1, p. 4, 2010.
[15] J. Yu, P. Lu, Y. Zhu, G. Xue, and M. Li, "Toward secure multikeyword top-k retrieval over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 10, no. 4, pp. 239–250, Jul./Aug. 2013.
[16] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. EUROCRYPT*. Berlin, Germany: Springer-Verlag, 2011, pp. 568–588.
[17] *NSF Research Awards Abstracts 1990-2003*. [Online]. Available: http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html, accessed 2004.
[18] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. EUROCRYPT*, 2004, pp. 506–522.
[19] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. TCC*, 2007, pp. 535–554.
[20] B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 262–267, Jan. 2011.
[21] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Efficient information retrieval for ranked queries in cost-effective cloud environments," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2581–2585.
[22] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 226–234.
[23] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 522–530.
[24] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, May 2000, pp. 44–55.
[25] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.
[26] D. Cash *et al.*, "Dynamic searchable encryption in very-large databases: Data structures and implementation," in *Proc. NDSS*, Feb. 2014.
[27] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.
[28] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2010, pp. 253–262.
[29] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–5.

HONGWEI LI (M'12) is currently an Associate Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, where he received the Ph.D. degree in computer software and theory, in 2008. He was a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, for one year until 2012. His research interests include network security, applied cryptography, and trusted computing. He serves as an Associate Editor of *Peer-to-Peer Networking and Applications*, a Guest Editor of *Peer to-Peer Networking and Applications* of the Special Issue on Security and Privacy of P2P Networks in Emerging Smart City. He serves on the Technical Program Committees for many international conferences, such as the IEEE INFOCOM, the IEEE ICC, the IEEE GLOBECOM, the IEEE WCNC, the IEEE SmartGridComm, BODYNETS, and the IEEE DASC. He is also a member of the China Computer Federation and the China Association for Cryptologic Research.



DONGXIAO LIU (S'14) received the B.S. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2013, where he is currently pursuing the master's degree with the School of Computer Science and Engineering. He serves as a reviewer of *Peer-to-Peer Networking and Application*. His research interests include cryptography, cloud computing security, and the secure smart grid.

YUANSHUN DAI (M'03) received the B.S. degree from Tsinghua University, Beijing, China, in 2000, and the Ph.D. degree from the National University of Singapore, Singapore, in 2003. He is currently the Dean of the School of Computer Science and Engineering with the University of Electronic Science and Technology of China, Chengdu, China, where he is also the Chaired Professor and Director of the Collaborative Autonomic Computing Laboratory. He has served as the Chairman of the Professor Committee with the School of Computer Science and Engineering since 2012, and the Associate Director at the Youth Committee of the National 1000 Year Plan in China. He has authored over 100 papers and five books, out of which 50 papers were indexed by SCI, including 25 IEEE TRANSACTIONS/ACM Transactions papers. His current research interests include cloud computing and big data, reliability and security, modeling, and optimization. He has served as a Guest Editor of the IEEE TRANSACTIONS ON RELIABILITY. He is also on the Editorial Boards of several journals.

TOM H. LUAN (M'13) received the B.Sc. degree from Xi'an Jiaotong University, Xi'an, China, in 2004, the M.Phil. degree from the Hong Kong University of Science and Technology, Hong Kong, in 2007, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2012. Since 2013, he has been a Lecturer in Mobile and Applications with the School of Information Technology, Deakin University, Melbourne, VIC, Australia. His research mainly focuses on vehicular networking, wireless content distribution, peer-to-peer networking, and mobile cloud computing.

XUEMIN (SHERMAN) SHEN (F'09) is currently a Professor and the University Research Chair of the Department of Electrical and Computer Engineering with the University of Waterloo, Waterloo, ON, Canada. He was the Associate Chair for Graduate Studies from 2004 to 2008. His research focuses on resource management in interconnected wireless/wired networks, wireless network security, and vehicular ad hoc and sensor networks. He served as the Technical Program Committee Chair of the IEEE VTC'10 Fall and the IEEE Globecom'07. He also served as the Editor-in-Chief of the *IEEE Network*, *Peer-to-Peer Networking and Application*, and *IET Communications*, a Founding Area Editor of the IEEE TRANSACTIONS ON Wbioscireless COMMUNICATIONS, and an Associate Editor of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and *Computer Networks*. He is also a registered Professional Engineer in Ontario, Canada, a fellow of the Engineering Institute of Canada and the Canadian Academy of Engineering, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and the Communications Society.