IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

Implementing Efficient Multi-Keyword Ranked Search Using Encrypted Storage Data

K. Anu[1], S. Sakthi Vinayagam[2], R. Raj kumar[3], G. Karthika[4]

PG Student, Computer Science, Chettinad college of engineering & Technology Karur, Karur, Tamilnadu[1, 4]

Assistant Professor, Computer Science, Chettinad college of engineering & Technology Karur, Karur Tamilnadu[2, 3]

Abstract: Cloud computing provide efficient storage to the user. The subcontracted records want to be encoded because the user wants to kept their data without leakage and trust. The encrypted data must be kept confidential. To overcome the difficulties we specify, by considering the big range of subcontracted files (statistics) within the cloud, we make use of the relevance rating and k-nearest neighbor strategies to expand an efficient multi-keyword search scheme which could go back the ranked seek consequences primarily based on the accuracy. inside this framework, we leverage an efficient index to in addition improve the search efficiency, and undertake the blind storage machine to conceal get right of entry to sample of the request user. safety evaluation demonstrates that our scheme can achieve confidentiality of documents and index, wormhole privateness, wormhole unmanagebility, and concealing get right of entry to sample of the request consumer. eventually, using widespread simulations, we show that our suggestion can attain a good deal improved efficiency in phrases of seek functionality and search time as compared with the prevailing proposals.

Key words: Cloud computing, searchable encryption, multi-keyword ranked search, 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 con dentiality 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 dif culties when other users need to access interested data with search, due to the dif culties 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 search-able 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 ef ciency for large databases with modest scari cation on security guarantees. Cao et al. [11] propose a multi-keyword search scheme sup-porting result ranking by adopting k-nearest neighbors (kNN) technique [12]. Naveed et.al. [13] propose a dynamic search-able 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 func-tions. First, the searchable encryption schemes should support multi-keyword search, and provide the same user experi-ence as searching in Google search with different keywords; single-keyword search is far from satisfactory by only return-ing 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 traf c by only sending back the most relevant results from cloud to search users. Third, as for the search ef ciency, 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 bene ts, most of the existing proposals, however, fail to offer suf cient 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 sum-marized 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 con-struct 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

 DOI 10.17148/IJARCCE.2016.53204

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 con-dentiality 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, respec-tively. 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 z. Both the encrypted documents C and encrypted index z are stored on the cloud server through blind storage system.

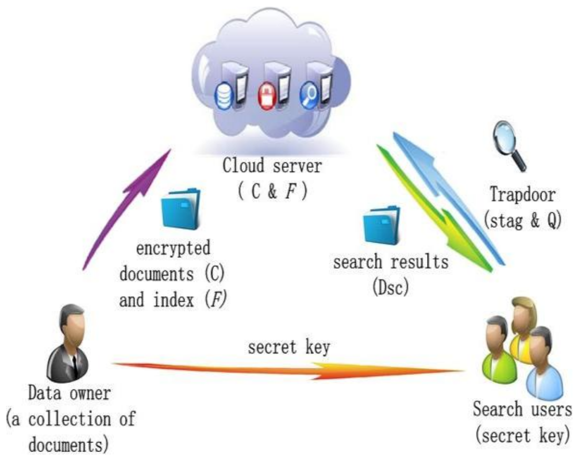


FIGURE 1. System model

When a search user wants to search over the encrypted documents, she rst receives the secret key from the data owner. Then, she chooses a conjunctive keyword set $ 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 z in the blind storage and computes the relevance scores with the encrypted query vector Q. Then, the cloud server sends back descrip-tors (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. Speci cally, the EMRS aims to provide the following four security requirements:

Con dentiality 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 know-ing 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. Speci cally, 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 Ef ciency: 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.

Con dentiality 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 weight-ing, term frequency $tf_{t;\ f}$ refers to the number of term t in a document f. Inverse document frequency is calculated as
idft D log N , where df^t denotes the number of documents df_t 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 con dentially 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 indi-cator 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 sup-port 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 xed-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. Speci cally, 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 struc-ture (usually an access tree) which de nes 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 z are both stored in the blind storage system, we would provide the general construc-tion 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 lever-age ciphertext policy attribute-based encryption (CP-ABE) technique in the EMRS. However, speci c construction of CP-ABE is out of scope of this paper and we only give a simple indication here. The notations of this paper are 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 Stor-age.

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
I	an efficient search index
$Enc_K()$	symmetric encryption algorithm using key K

A. SYSTEM SETUP

The data owner takes a security parameter , and outputs two invertible matrixes M_1; M_2 2 $R^{(dC2)\ (dC2)}$ as well as a (dC2)-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 9, a pseudorandom generator 0 and a hash function 8Vf0; 1g ! f0; $1g^{192}$. 9 and 0 are based on the AES block-cipher [13]. Then, the data owner chooses a number > 1 that de nes the expansion parameter and a number that denotes the minimum number of blocks in a communication.

1) B.KEYGEN

The data owner generates a key K_9 for the function 9 and sends it to the search user using a secure channel.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

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 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 rst 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 $_i$ D 9_{K9} (id_i) as the input of the function 0. Generate a suf ciently long bit-number through the function 0 using the seed I and parse it as a sequence of integers in the range [nb]. Let [i;l] denote the rst l integers of this sequence. Generate a set S_f D [$_i$; max(d $size_i$e;)].

Step 2: Let S_f^0 D [$_i$;], 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^0 S_f that contains $size_i$ integers, and make sure that the blocks indexed by these integers in the subset S_f^0 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 rst $size_i$ integers indexing free blocks and make these integers form the subset S_f^0 . Mark these blocks as unfree. Then, write the document d_i to the blocks indexed by the integers in S_f^0 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 speci c 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 xed-size blocks indexed by the integers, that are generated by applying the seed $_i$ to the pseudorandom generator 0. 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 $size_i$ integers as the set S_f . Here the choice of the parameter 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 D (p_1; p_2; p_d) for each document using the TF-IDF weighting technique, where p_j for j 2 (1; 2 d) represents the weighting of keyword $!_j$ in document d_i. Then, the data owner extends the p to a (dC2)-dimension vector p . The (dC1)-th entry of p is set to a random number " and the (dC2)-th entry is set to 1. We would let " follow a normal distribution N (; ^{2}) [11]. For each document d_i, to com-pute 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 (dC2)-dimension vectors p^0 and p^{00} using the vector S. For the j-th item in p, set

$$p_j^0 \text{ D } p_j^{00} \text{ D } p_j \text{ ; } \quad \text{if } S_j \text{D1} \qquad (1)$$
$$p_j^0 \text{ D } \frac{1}{2} p_j \text{ C } r; \quad p_j^{00} \text{ D} \frac{1}{2} p_j \quad r; \qquad \text{otherwise}$$

where S_j represents the j-th item of S. Then compute the P D fM_1^T p^0; M_2^T p^{00}g 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 rst 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 8(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 rst block of the document d_i with index number j is as

$$Enc_{(K_i 8(j))}(H\ (id_i) jj size_i jj data) \quad (2)$$

And the rest of the blocks of d_i is as

$$\text{Enc}_{(K_i \ 8(j))}(\text{H } (id_i) \text{jjdata}) \qquad (3)$$

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 z. First, the data owner de nes the access policy $_i$ for each document d_i. We denote the result of attribute-based encryption using access policy $_i$ as ABE $_i$ (). The data owner initializes z to an empty array indexed by all keywords. Then, the index z can be constructed as shown in Algorithm 1.

Algorithm 1 Initialize z

```
for each keyword ! 2 W do
    Set t an empty list
    for each document d_i containing the keyword ! do
        Get the associated vector P of d_i
        Choose a random number x
        Dsc   ABE_i (id_i jjK_i jjx)
        Append the tuple (Dsc; P) to t
    end for
    z[!] D t
end for
return z
```

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

As we can see, the index z maps the keyword to the encrypted relevance vectors (P) and the descriptors (Dsc) of the documents that contain the keyword. And each list z[!] can be transformed to be stored in the blind storage system with ! as the document id. Speci cally, for each z[!], the data owner computes $_!$ D 9_{K9} (!) as the seed for the function 0 to generate the set S_f . Here, for each block of z[!] indexed by the integer j, the data owner adds an encrypted header as $Enc_{(KT\ 8(j))}$(H (!)jjsize$_!$), where size$_!$ represents the number of blocks that belong to z[!]. Finally, the data owner writes the index z 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 ! to compute the seed for each z[!]. Moreover, each header of the blocks of the documents contains the encrypted H (id_i) and the rst block indicates the size$_i$. And the blocks of index z are different from those of the documents. Each header of the blocks of index z is denoted as $Enc_{(KT\ 8(j))}$(H (!)jjsize$_!$). 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 z[!].

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 \$ D ($!_1$; $!_2$; $!_l$) 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 $!_j$ 2 \$ or not. Then, the search user chooses two random numbers r, t and scales the query vector q to a (dC2)-dimension vector q as

$$q \ \ D\ frq; r; tg \qquad (4)$$

Then, the search user chooses a random number r^0 and splits the vector q into two (dC2)-dimension vectors q^0 and q^{00}. For the j-th item in q , set

$$q_j^0 \ \ D\ q_j^{00}\ D\ q_j\ ; \quad \text{if } S_j\ D\ 0 \qquad (5)$$

$$({}_{qj}0 \ \ D\ {}_2{}^1 q_j\ C\ r^0; \quad q_j^{00}\ D\ {}_2{}^1 q_j \quad r^0; \qquad \text{otherwise}$$

The search user computes the Q D fM_1 ^{1} q^0; M_2 ^{1} q^{00}g as the encrypted query vector.

Step 2: The search user chooses the estimated least frequent keyword $!^0$ in the conjunction \$ and computes the seed $_!0$ D 9_{K9} ($!^0$). Then the search user generates a long bit-number through the function 0 using the seed $_!0$. The search user chooses the sequence [$_!0$;] and randomly adds dummy integers to the sequence. The search user down-loads the blocks indexed by these 2 integers and decrypts the header using the key K_T 8(j), where j is the index number of the block, to nd the rst block of the list z[$!^0$], which consists of the descriptors and the encrypted relevance vectors of the documents containing $!^0$. Then the search user obtains the size$_!$0 from the rst block and computes the set $S_!$ D [$_!0$; size$_!$0]. The search user randomly adds size$_!$0 dummy integers to the set $S_!$ resulting in a set $S_!^0$ of 2 size$_!$0 integers. And the extended set $S_!^0$ 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 z in the blind storage and retrieves the blocks indexed by the integers to obtain the tuples (ABE $_i$ (id_ijj$K_i$jjx); P) on these blocks. Note that, these blocks consist of the blocks of z[$!^0$] 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:

$$\begin{aligned} Score_i\ D\ P\ \ Q \\ D\ fM_1{}^T\ \ p^0; M_2{}^T\ \ p^{00}g\ fM_1\ {}^1\ \ q^0; M_2\ {}^1\ \ q^{00}g \\ D\ p^0\ \ q^0\ C\ p^{00}\ \ q^{00} \\ D\ p\ \ q \\ D\ (p; ; 1)\ \ (rq; r; t) \\ D\ r(pq\ C\ ")\ C\ t \qquad (6) \end{aligned}$$

Finally, after sorting the relevance scores, the cloud server sends back the descriptors ABE $_i$ (id_ijj$K_i$jjx) 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 $_i$ (id_ijj$K_i$jjx), 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 $_i$ D 9_{K9} (id_i) for the function 0. Generate a suf ciently long bit-number through the func-tion 0 using the seed $_i$, parse it as a sequence of integers in the range [n_b] and choose the rst integers as the set S_f^0. Retrieve the blocks indexed by these 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 8(j), until she nds the rst block of the document d_i. If she does not nd the rst 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 rst block.

Step 3: Then, the search user computes l D d size$_i$e. If l , compute S_f D [$_i$;]. Otherwise, compute S_f D [$_i$; 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 con dentiality 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 con dentiality, 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 z 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 $_i$ (id$_i$jjK$_i$jjx) to get the document id and the associated symmetric key. Thus, the con dentiality 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 9 and 0 to help the search user compute keyword-related token stag using the secret key K_9 . Without the secret key M_1; M_2; S and K_9 , 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 de ned 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 de ned as fM_1 ^{1} q^0; M_2 ^{1} q^{00}g in the EMRS. First, the search user needs to extend the query vector q to q . As we can see, the (dC1)-th and (dC2)-th entry of the vector q are set to random values r and t. So there are 2^r 2^t possible values, where the number r and t are $_r$ -bit or $_t$ -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 D 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 and each dimension of the vector q^0 is $_q$-bit long. We can see that $_r$, $_t$, and $_q$ are independent of each other. Then we can compute the probability that two encrypted query vectors are the same as

$$PD \frac{1}{2^{r}2^{t}2^{q}} D \frac{1}{2^{r C t C q}} \tag{7}$$

Therefore, the larger these parameters are, the lower the prob-ability 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 rst obtains the size$_!$ from the cloud server using the sequence of 2 integers, half of which are dummy integers. Then, the search user computes the set $S_!$ D [$_!$; size$_!$] and adds size$_!$ dummy integers to the set $S_!$ to form the stag. Thus, each stag contains 2 size$_!$ random integers, half of which are random integers. Suppose the integers are n_b bits long. Then the probability that the two stags are the same is

$$P^0 D \frac{1}{2^{2\ size_! n_b}} \tag{8}$$

Hence, if we choose 12-bit long n_b, 3-bit long extension parameter and size$_!$ is supposed to be 8-bit long, the probability P^0 < $\frac{1}{2^{576}}$, 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 fre-quency of the keywords and deduce some information using the statistic knowledge in [10] and [13].

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

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 perfor-mance by the choice of parameters. We de ne 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 , and to measure the P_{err} . We denote D 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. is the ratio that scales the number of blocks a document contains to the number of blocks in the set S_f. is the minimum number of blocks in a transaction. Then, according to [13], we can compute the P_{err} as

$$P_{err}(;\ ;) \quad \max_{X} {}^{n\ 1} \ d\,ne \ ! \ \frac{1}{\quad} \ i \ \frac{1}{\ } \ d\,ne\,i \qquad (9)$$

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 in uence 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 guaran-tees 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 speci c 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 simula-tions 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 perfor-mance 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 ef ciency.

1) INDEX CONSTRUCTION

Index construction in the EMRS consists of two phases: encrypted relevance vector computation and the efficient index z construction via blind storage.

As for the computation of encrypted relevance vector, the data owner rst 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 in uence 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 C 2) (d C 2) matrix and a (dC2)-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

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

subindex of one document is xed. 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.

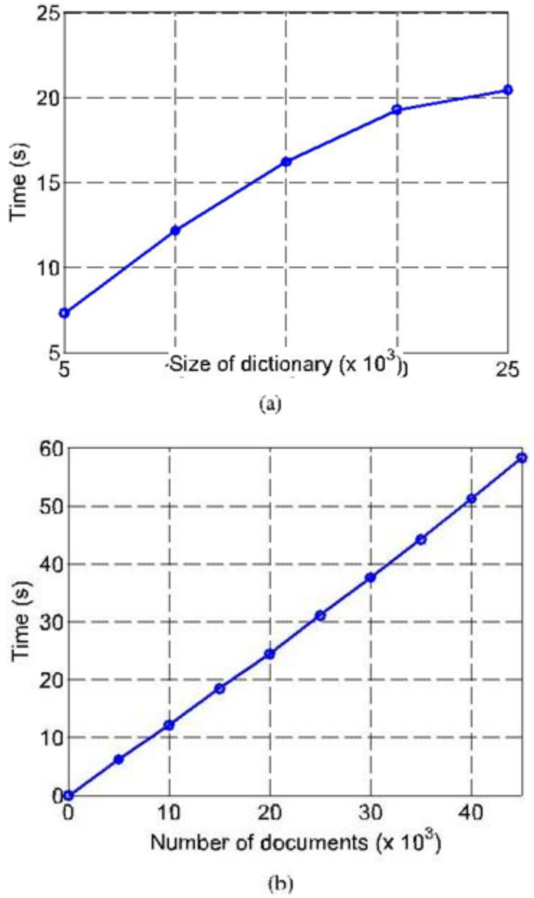


FIGURE 2. Time for calculating relevance score. (a) For the different size of dictionary with the same number of documents, m D **10000. (b) For the different number of documents with the same size of dictionary,** j**W**j D **10000.**

Finally, we adopt the index z via the blind storage in the EMRS to improve search ef ciency and conceal the access pattern of the search user. For each keyword ! 2 W , we need to build the list z[!] of tuples (ABE $_i$ (id$_i$jjK$_i$jjx); P) of documents that contain the keyword and upload it using the B.Build function. So the computation complexity to build the index z is O(%d), where % represents the average number of tuples contained in the list z[!] and is no more than the number of document m. Since the access pattern is not con-sidered in most schemes, we are not going to give the speci c 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 (9 and 0) to generate a sequence of random integers. Compared with time cost to compute the encrypted query vector which is linearly increasing with the size of the keyword dictionary, time cost for computing stag is negligible.

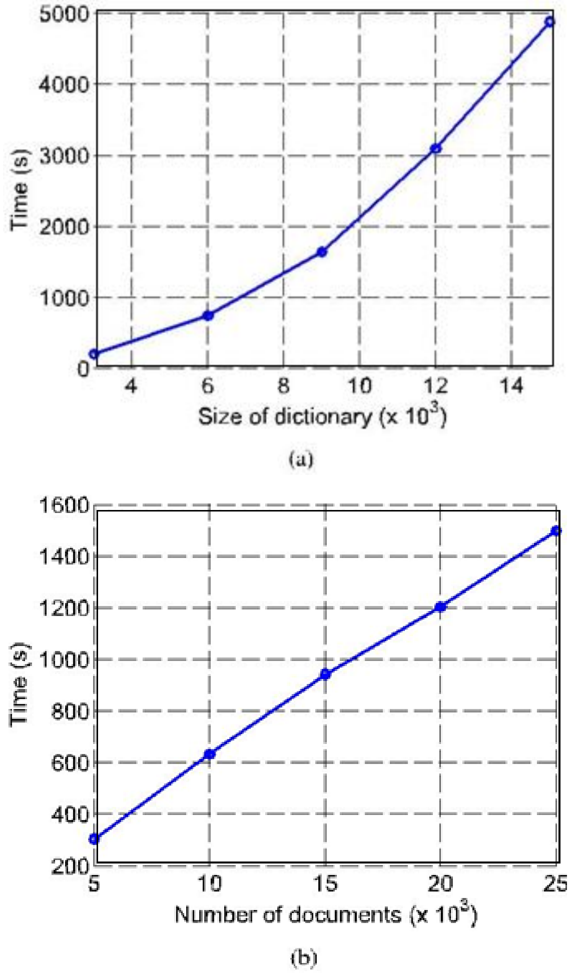
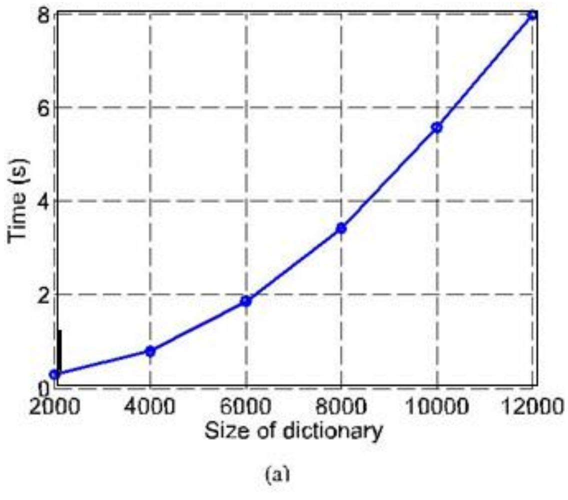


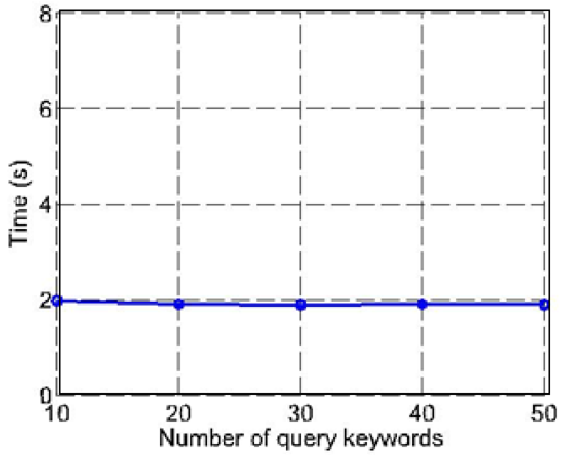
FIGURE 3. Time for computing the encrypted relevance vectors. (a) For the different size of dictionary with the same number of documents, m D **6000. (b) For the different number of documents with the same size of dictionary,** j**W**j D **4000.**



DOI 10.17148/IJARCCE.2016.53204

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016



(b)

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, j$j D **20. (b) For the different number of query keywords with the same size of dictionary,** jWj D **6000.**

As for computing the encrypted query vector Q, the search user needs to compute two multiplications of a (d C 2) (d C 2) matrix and a (dC2)-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 (dC2)-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 z 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 signi cantly 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 (dC2)-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($\%_s$d), where $\%_s$ represents the the number of documents which contain the keyword applied by the keyword-related token stag and the 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 $\%_s$ 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 signi cantly 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 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.

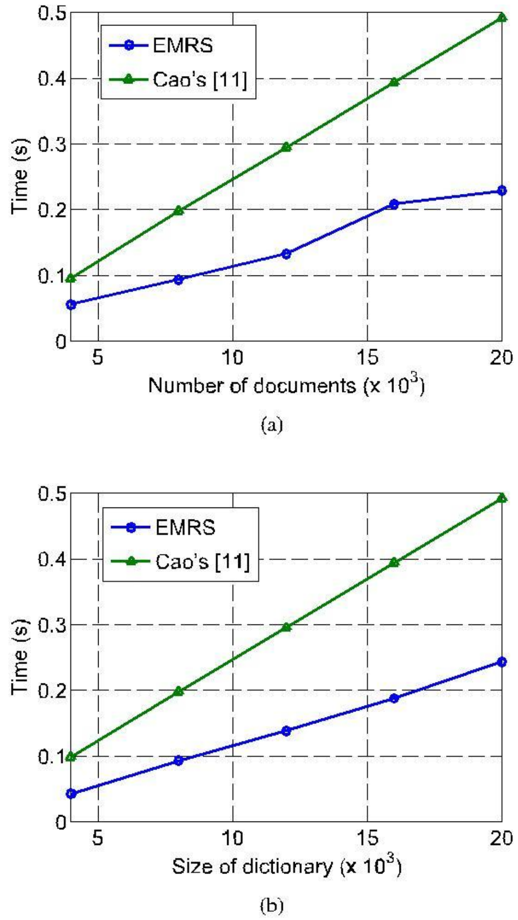


FIGURE 5. Time for search on the cloud server. (a) For different number of documents with the same size of keyword dictionary and number of searched keywords, jWj D **8000,** j$j D **20.**
(b) For different size of keyword dictionary with the same number of documents and searched keywords, m D **8000,** j$j D **20.**

C. COMMUNICATION OVERHEAD

When the system is once setup, including generating encrypted documents and index, the communication overhead is mainly in uenced 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

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

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 C 2) $_q$, where d represents the size of the keyword dic-tionary and each dimension of the encrypted query vector is $_q$-bit long. According to Cash's scheme [10], when a search user wants to query over the cloud server using a conjunctive keyword set \$, she needs to compute stag for the estimated least-frequent keyword and xtokens for the other keywords in the set \$. And, each xtoken contains j\$ j 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 xtokens is linear to %, the number of documents containing the keyword related to the stag. This results in much unnecessary communication overhead of %j\$ jjGj, where jGj 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$_!$ and then obtain the results. This requires one or two round communication of size$_!$ n_b bits, where is the extension parameter, size$_!$ is the number of blocks of documents containing !, 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 ef ciency 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	$\varrho\lvert\varpi\rvert\lvert G\rvert$
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 communica-tion 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 speci c deploy-ment 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, and the larger these two numbers are, the higher security guar-antee the scheme provides, as we discussed in Section V. The size of returned results for each document can be a size$_!$ 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 classi ed into two types: Searchable Public-key Encryption. (SPE) and Searchable Symmetric Encryption (SSE).

Boneh et al. [18] rst 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 con-junctive, 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 lever-aging 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 function-alities, SPE are not efficient since SPE involves a good many asymmetric cryptography operations. This motivates the research on SSE mechanisms.

The rst 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 de nitions 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

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

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 sup-ported in [26]. Naveed et.al. [13] construct a blind storage sys-tem 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 con dentiality 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 ef - ciency 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 net-works," *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 pro-tecting 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 search-able 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. Ro³u, 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 ne-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: Veri able 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 de nitions 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.