IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2018, accepted June 4, 2018, date of publication June 8, 2018, date of current version June 29, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2845468

PRMS: A Personalized Mobile Search Over Encrypted Outsourced Data

QIANG ZHANG[1], QIN LIU[2], AND GUOJUN WANG[3], (Member, IEEE)
[1]School of Information Science and Engineering, Central South University, Changsha 410083, China
[2]College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[3]School of Computer Science and Educational Software, Guangzhou University, Guangzhou 510006, China

Corresponding author: Guojun Wang (csgjwang@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61632009 and Grant 61472451, in part by the Guangdong Provincial Natural Science Foundation under Grant 2017A030308006, in part by the High-Level Talents Program of Higher Education in Guangdong Province under Grant 2016ZJ01, in part by the Fundamental Research Funds for the Central Universities of Central South University under Grant 2017zzts141, in part by the Hunan Provincial Education Department of China under Grant Number 2015C0589, and in part by the Hunan Provincial Natural Science Foundation of China under Grant 2015JJ3046.

ABSTRACT At present, searchable encryption is a very promising direction in the field of cloud computing. However, most existing works focus on keyword-based search schemes and regardless of personalized search needs, a keyword-based search does not take into account the user's location information and cannot exactly match users' search intentions. Since location information is very important in mobile searches, we propose a personalized mobile search (PRMS) over encrypted outsourced data, and we convert the user's location information into distance information to generate the user location model, forming a location query matrix with user location information. The matrix is then used to encrypt the user's location query and conceal the location information in the location query matrix. In this paper, we combine a user's interest preference and location information in personalized searches over encrypted outsourced data, adopt the law of universal gravitation to calculate scores of files in the cloud, and return the first K results with the highest gravitational forces. For the first time, we propose a PRMS-improvement scheme that is applied to an encryption method to build the content index and location index, which can greatly reduce the time for model construction. Through the PRMS scheme, we realized personalized mobile searches based on user's interests and location information.

INDEX TERMS Searchable encryption, cloud computing, personalized search, the law of universal gravitation, privacy-preserving.

I. INTRODUCTION

With the popularity of mobile phones, location-based [1], [2] mobile search is becoming more and more important. To protect the privacy of data, people usually choose to upload data to the cloud after encrypting it. However, data encryption makes utilization of the data more difficult. Moreover, a major problem in mobile searches is that the device's screen is too small to read easily. To achieve this objective of highly relevant results, an effective method to find information quickly and accurately over encrypted outsourced data is needed.

For the same keyword queries, different users have different goals when searching for information. In mobile scenarios, due to the small screen size of the mobile phone and limited battery power, users are more in need of context-aware personalized search results. In order to better understand personalized mobile search, we consider the following application scenario. A user arrives in Beijing for the first time and wants to find a hotel nearby. He enters the keyword "hotel" and submits the query to obtain the results. In such a scenario, location-based search results are needed of the most popular or nearest hotel. Therefore, results are not helpful for users if they only consider the user's query information and ignore the location information. Moreover, taking into account the user's financial abilities, he may need a five-star or economic hotel. The purpose of the personalized mobile search is to place the results that the user most wants in the key position, so that he can quickly obtain the results.

The user model reflects personal preference and determines the accuracy of the personalized search. Fu *et al.* [3] builds the user model through the search history and obtains

personalized search results with user preferences through the user model.

We sort the search results by the model of the user to obtain personalized search results. The degree to which the trapdoor matches the files in the cloud server determines the order of the files returned. There are three problems in personalized searches:

1. How to incorporate user interests and situational information into user models.
2. How to calculate the degree of matching between the user's true search intent and the file in the cloud server.
3. How to securely compute the gravitational force between the user query and each file, then return personalized sorting results to users.

For the first problem, many existing personalized searches mine users' preferences through the historical click data [4], [5]. In order to return highly relevant results to the users, Leung *et al.* [6] describes the user model using content and location concepts. The user's interests and location information can provide a more accurate user model. For the second problem, personalized search is a good choice to understand the user's search intention. Du *et al.* [7] noted that the user needs the results that match both his query and interests. The final score of personalized search was related to two factors: the user's query and the user interest model. The top-k results are then returned to the user according to the final score. For the third problem, Cao *et al.* [8] proposed a scheme that can achieve multi-keyword searches in ciphertext while protecting users' privacy. It uses ''secure inner product similarity'' to calculate how many keywords match a file. However, they consider the different keywords are equally important and the search is not accurate enough.

Moreover, the user's interest information has different levels of importance to the user's location information. Location information is particularly important in location-based mobile searches. We present the personalized mobile search (PRMS) scheme, which has two types of information: location and content. By introducing the location information that offers PRMS an additional dimension for personalized search, it can enhance search quality for users and improve the search experience.

In this paper, we proposed a PRMS over encrypted outsourced data. The contributions of our work are as follows:

- 1) We learn from the law of universal gravitation and use it to sort search results for personalized searches. The PRMS scheme enables users to conduct personalized searches, which are not only based on user interests, but also on the location information. Our PRMS scheme greatly enhances the accuracy of location-based searches, allowing users to quickly obtain the needed information.
- 2) For the first time, we propose a PRMS_improvement scheme that is applied to an encryption method to build the content and location indexes. Our experiments show that this method can greatly reduce the time for model construction.
- 3) By converting the user's location information, we protect the user's location privacy while obtaining context-aware, personalized search results.
- 4) Through the experiments on the Yelp dataset, we prove that our PRMS scheme is efficient and feasible. The cloud server only needs to return the top K results that significantly reduce the communication overhead.

II. RELATED WORK

A. SEARCHABLE ENCRYPTION

Searchable encryption technology has emerged to retrieve information in the ciphertext environment [9]. Song *et al.* [10] encrypts the keywords using the stream cipher. By individually matching the keywords with the ciphertext file, it can learn whether the keywords are included in the ciphertexts. This work is a new chapter for the keyword searches in ciphertext environments. Subsequently, Chang and Mitzenmacher [11], Curtmola *et al.* [12], Liu *et al.* [13] and others proposed many improvement schemes that injected new vitality into the searchable encryption. Wang *et al.* [14] use the keyword frequency to sort the search results, and found the same keywords return different search results while protecting privacy. Boneh *et al.* [15] present the first keyword search of the public-key encryption scheme, which was used to show the routing problem of the server is not credible. In this scheme, users only need a private key that can search data that is encrypted by the corresponding public key. However, these schemes only support single-keyword searches over encrypted outsourced data. To solve this problem, researchers [16]–[18] proposed a variety of conjunctive keyword searches in the encrypted data. Cash *et al.* [19] and Stefanov *et al.* [20] presented the problem of privacy disclosure in searchable encryption schemes. After that, fuzzy search [21] was proposed to solve the problem of user spelling mistakes. Single-keyword [14] and multi-keyword [8] ranking search schemes ensure the security of data and prevent the leakage of privacy information. Later, Fu *et al.* [3] proposed a multi-keyword personalized search scheme in encrypted data. However, most searchable encryption schemes can't keep up with the development of mobile searches because they ignore the importance of location information and efficiency is not high enough.

B. PERSONALIZED SEARCH

Personalized searches have received increasing attention in recent years, including whether the user model can fully reflect all information of the user (including user's interest and location) that directly affects the accuracy of the search results [22], [23]. Therefore, how to build a user model is a subject worth studying. The user submits the query, which means that he wants to obtain results related to the query, so the returned results should not only be related to the user model, but also the user query. There is much research that first returns relevant results based on the user's query, then reorders the query results according to the user model, so as

to achieve the personalized search [24]–[27]. Du *et al.* [7] not only builds the user model with the user's interest tag, but also integrates the dislike tag into the user model, and thus obtains more accurate personalized search results. However, many personalized search schemes are only applicable to the plaintext environment, and there will be nothing to do in the ciphertext environment. Fu *et al.* [3] realizes the personalized search in the ciphertext environment by constructing the user's model and using the safe inner product calculation method. Users can directly obtain personalized search results without re-ranking search results. However, this work did not consider the user's location information, making it unsuitable for location-based mobile searches, and the index construction is too inefficient to be used for large-scale data searches.

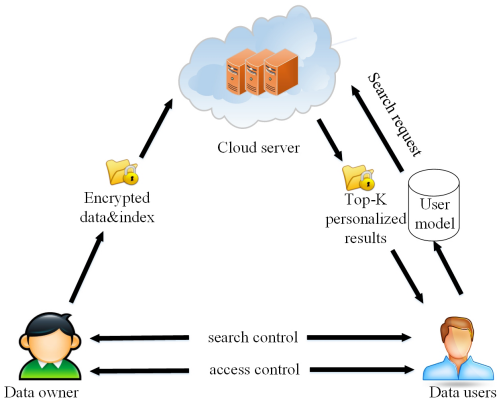

FIGURE 1. Architecture of the search over encrypted cloud data.

III. PROBLEM FORMULATION

A. SYSTEM MODEL

As shown in Figure 1, the system model consists of three types of entities: the data user, the data owner, and the cloud server. Unlike previous work [8], [14], [21], [28]–[31], the index consists of the content index and the location index. Our index construction is quite efficient, because we aggregate all file vectors into a matrix, and the encrypting procedure only needs to be completed once regardless of the number of files. The user model consists of the user interest model and the user location model, which are stored on the client side, in order to protect the privacy of the user. The user location model is built upon the location of the user and the *location* information in the database. The query is obtained by query reformulation through the user model. Finally, the encrypted query will be sent to the cloud. After obtaining the user's query, the cloud server searches the index, and then returns the top K encrypted files with the highest relevancy score to the user. In this paper, we use the law of gravitation to calculate the relevance of the score between the files and the user's query.

B. NOTATIONS

- F-the collection of plaintext files, each of which is associated with a *location* point, with latitude and longitude information, $F = (F_1, F_2, \ldots, F_{m_f})$.
- C-the collection of encrypted files, denoted as $C = (C_1, C_2, \ldots, C_m)$.
- q-the query content matrix, where the value of each location represents the weight of the keyword.
- p-the file matrix, the plain index for F, where each line represents the index of a file.
- R-used to describe characteristics of resources, each value represents the characteristic of a file (e.g. the user's average score for a restaurant).
- q_l-the location matrix, which is the user location model for a query of the user.
- p_l-the resource location matrix, which is based on R.
- I_C-the encrypted content index, which is based on p.
- Q_C-the encrypted query content matrix, which is based on q.
- Q_L-the encrypted user location matrix, which is based on q_l.
- I_L-the encrypted location index, which is based on p_l.
- F_g-the gravitational force between the query and the file.
- $file(1:K)$-the top-K files according to the gravitational force F_g.

C. THREAT MODEL

Honest-but-Curious (HBC) [8], [32]–[34]: In this model, an attacker strictly follows the entire protocol, but for some purposes, mines sensitive information from the known information (for example, inferring the user's income level through online shopping records or inferring the user's home address through location information), where the cloud server is honest and curious. Since the concealed security is dangerous and stupid, we assume that the cloud server not only knows the ciphertext, but also the encryption and decryption algorithms. In order to better evaluate the security, we divide it into three levels:

- Level 1: The attacker knows the encrypted files C, the encrypted index I, and the encrypted query matrix T.
- Level 2: The attacker not only knows C, I, and T, but also knows some of the plaintext indexes P_A.
- Level 3: The attacker not only knows C, I, and T, but also knows some of the plaintext indexes P_A and those corresponding encrypted values I_A.

D. DESIGN GOALS

- Personalized mobile search: The PRMS scheme is mainly used to solve the personalized search in mobile environments. In order to be more suitable for mobile search environments, the personalized search results are not only related to the user's personal interest, but also related to the location information of the user.
- Location privacy protection: As a personalized search scheme in mobile environments, PRMS needs to use the location information of a user. However, the location information is the user's sensitive information. Therefore, the PRMS scheme not only needs to provide a personalized search, but also protect the user's location privacy.

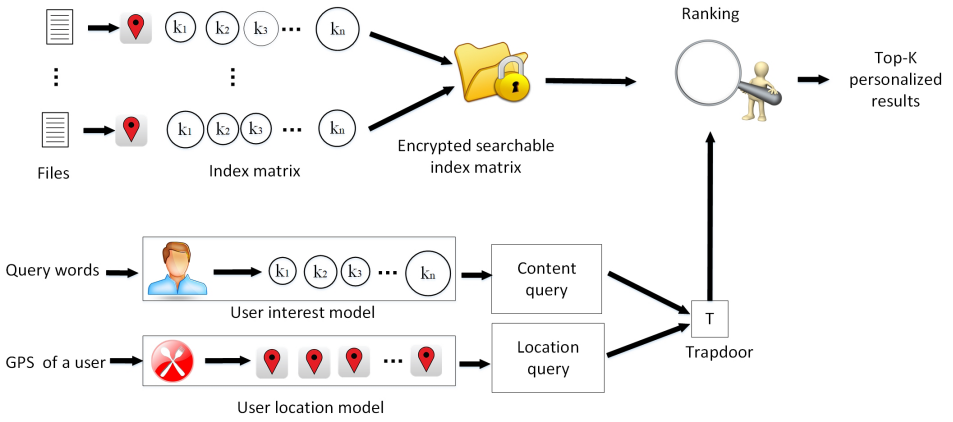

FIGURE 2. Overview of PRMS scheme.

- Other privacy-preserving: Data owners and users encrypt files, indexes, and queries to protect data and their privacy. The cloud server shouldn't mine sensitive information through encrypting files, indexes, and user queries.

E. DEFINITION

1) KEYWORD WEIGHT

The weight of each keyword in a file represents the file relevant to the keyword. We adopt the widely used method of vector space modeling to represent file profiles.

In the vector space model (where queries, files, and users are all mapped to keyword vectors in a universal term space [35]), the index constructed by "$TF \times IDF$" method can well reflect the importance of keywords in files [36]–[38]. This paper uses "$TF \times IDF$" to build the index of the file so that the keywords can better reflect the file.

2) GRAVITATIONAL FORCE

There exists gravitational force between any two objects. The magnitude of gravitational force represents the degree of mutual attraction between two objects. In the paper, Gravitational force $F_g(i, :)$ is represents the relevance of the query to a file. The higher the score, the greater the relevance of the query to the file.

3) SECURE INNER PRODUCT

We use two matrices (the query matrix and index matrix) to calculate the gravitational force of the query to a file. However, if we directly compute the inner product of two matrices on the cloud server, it has the risk of disclosure of privacy. The secure inner product [8], [39] is adopted in our scheme. The algorithm computes the inner product of two encrypted matrices $E(p)$ and $E(q)$ without knowing the actual values of p and q possible. We can obtain $E(p) \cdot E(q) = p \cdot q$.

IV. THE BASIC DESIGN

A. OVERVIEW

As shown in Figure 2, our scheme adopts "secure inner product" like PRSE [3] to calculate the gravitational force between the query and a file. In the PRMS, the user model includes two parts: the information of the user's location, which is obtained from mobile devices, such as GPS; and user interest on keywords, which is obtained from the user's search history. Therefore, in our scheme, the search results not only relate to the keywords of the query, but also the user's historical interest and location.

B. FRAMEWORK

The PRMS scheme contains the following algorithms:

- **Setup**: The data owner randomly generates a key as SK. To reduce the overhead of encrypting and decrypting files, the data owner uses symmetric cryptography (such as 3DES, AES) to encrypt each file in the fileset F.
- **BuildIndex**(F, SK): The data owner builds the content index and location index of files F, which are encrypted by the key SK.
- **BuildUM**: The user model is generated on the client side, which is divided into two parts: the user interest model and the user location model.
- **GenQuery**(q, U, SK): According to the query q of the user and the user model U, we obtain the new query of the user, which reflects the user's search intent, then generate the corresponding trapdoor T, which is encrypted by the key SK. Finally, T and K (K is the number of search results the user wants to return) will be uploaded to the cloud in order to obtain results that the user most wants.
- **Ranking**(I, T, K): The cloud server calculates the value of gravitational force between the trapdoor T and the index I based on the safe inner product, and it will return K encrypted files with the highest gravitation force to the user.

C. USER MODEL

The more accurate the user model, the more accurate the personalized search results obtained by the user. The PRMS scheme combines the user's interests and location information, making it more suitable for location-based mobile searches.

1) USER LOCATION MODEL

The client stores the type of the *location* represented by each file in the resource and the location information $\{type, (GPS_{ix}, GPS_{iy})\}$. *type* can anything such as hotel or restaurant, to name a few. When the user submits the query keyword, the user obtains the current location (GPS_{ux}, GPS_{uy}) through the mobile device.

We calculate the distance between each *location* and the user' location, and then set the weight of the *location* to be inversely proportional to the square of the distance. The user location model based on GPS information is generated at the client. Therefore, this process does not reveal the user's location privacy.

FIGURE 3. The weight of *location* keywords in user model.

When a user submits a query, the user's location is shown in Figure 3. The PRMS scheme obtains the user's location by reading the GPS information. The weight of the "Dragon wall restaurant" in Figure 3 can be calculated as follows:

$$q_l(i) = \frac{1}{d(i)^2} \tag{1}$$

where $q_l(i)$ is the weight of the "Dragon wall restaurant" in this query, and $d(i)$ is the distance between the user and the "Dragon wall restaurant" when the user submits the query, which can be calculated using the GPS information of both.

A user location model of user i is denoted by U_{il}.

$$U_{il} = (l_1 : q_l(1), l_2 : q_l(2), \ldots, l_m : q_l(m)) \tag{2}$$

Let the value represent the user location model. We obtain a user location model as follows:

$$q_l = (q_l(1), q_l(2), \ldots, q_l(m))$$

The details of the build are described in Algorithm 1.

Algorithm 1 Built of User Location Model

Input: (GPS_{ux}, GPS_{uy}): the GPS information of a user; (GPS_{ix}, GPS_{iy}): the GPS information of a *location*, e.g. a restaurant.

Output: q_l.

1: **for** $i = 1 : m$ **do**
2: $\quad d(i) = \frac{distance(GPS_{ux}, GPS_{uy}, GPS_{ix}, GPS_{iy})}{180 * pi * 6370}$;
3: $\quad q_l(i) = \frac{1}{d(i)^2}$;
4: **end for**
5: **return** q_l.

2) USER INTEREST MODEL

To return the search results that best match the user's search intent, we construct a user's interest model, and Leung *et al.* [6] captures user preferences by mining user click data. Du *et al.* [7] constructed a multi-level user model based on the user's preference to make it fully reflect the real needs of users. In order to return the ciphertext that is in accordance with the user's interest, this paper uses the method of literature [3] to build a user interest model, which constructs a user interest model with semantic information through the user's query history and WordNet [40] English Vocabulary Database.

A user interest model of user i is denoted by U_{ic}:

$$U_{ic} = (k_{i,1} : m_{i,1}, k_{i,2} : m_{i,2}, \ldots, k_{i,n} : m_{i,n}) \tag{3}$$

Let the value represent the user interest model. We obtain a user interest model as follows:

$$U_{ic} = (m_{i,1}, m_{i,2}, \ldots, m_{i,n}) \tag{4}$$

A scoring mechanism was adopted to build a user interest model. Whenever a user submits a query, the PRMS scheme uses the user's query to update the user interest model. For example, when the user's query matrix $q = (1, 1, 0, 0, 0, 0, 1, 0, 0)$, the user interest model $U_{ic} = (9, 0, 8, 1, 0, 0, 2, 0, 7)$, The third keyword is synonymous with the second keyword, and the similarity of the two keywords is 0.4. Then, the updated user interest model is $U_{ic} = (10, 1, 8.4, 1, 0, 0, 3, 0, 7)$.

V. CONSTRUCTION OF THE SCHEME

In the PRMS scheme, we integrate the user's preference and location information into the user's query and return the first K most relevant ciphertexts to the user by calculating the gravitational force between the user query and each file index. The scheme is particularly suitable for location-based mobile searches to improve the user's search experience.

A. SETUP

The data owner randomly produces an $(n+tc)$-bit vector as s_1, an $(m + tl)$-bit vector as s_2, two $(n + tc) \times (n + tc)$ invertible matrices $\{M_1, M_2\}$ and two $(m + tl) \times (m + tl)$ invertible matrices $\{LM_1, LM_2\}$. Therefore, the secret key SK is the 6-tuple $\{s_1, M_1, M_2, s_2, LM_1, LM_2\}$.

B. BuildIndex(F,SK)

To obtain personalized mobile search results in encrypted files, the PRMS scheme needs to establish an index of files. In the paper, we divide the file index into two parts: the content index based on the file content and the location index based on the *location* of the file. Next, we encrypt content index and location index to protect the security of data and the user's query privacy.

We first build the file content index. The data owner uses the "$TF \times IDF$" method to build a "double" data structure p as the content index, where $p(i,:)$ is the index of file F_i, and $p(i,:)$ is a $1 \times n$ matrix. Next, we expand every index $p(i,:)$ into $(n+tc)$ dimensions as $p(i,:)$, where the $(n+g)th$ ($g \in [1, tc]$) set the same random number during the dimension extension. Then, the splitting procedure is the effect on $p(i,:)$ and splits it into two random row matrices, denoted by $\{p'(i,:), p''(i,:)\}$. The s_1 function is a splitting indicator. If $s_1(j)$ is equal to 0, both $p'(i,j)$ and $p''(i,j)$ are equal to $p(i,j)$; if $s_1(j)$ is equal to 1, and $p'(i,j)$ and $p''(i,j)$ are random values while their sum is equal to $p(i,j)$. The split data is encrypted as $I(i,:) = [p'(i,:) * M_1^T, p''(i,:) * M_2^T]$. After all files are encrypted, the data owner obtains $I_C = [p' * M_1^T, p'' * M_2^T]$.

After building the content index, we build the location index as follows:

The data owner builds an $m \times m$ diagonal matrix p_l for the location index, where every diagonal element is a characteristic of the corresponding file (such as restaurant rating, p_l=diag(R)). Next, the plaintext index $p_l(i,:)$ is extended from m dimensions to $(m+tl)$ dimensions as $p_l^\star(i,:)$. These processes are similar to the content index construction and the $(m+g)th$ ($g \in [1, tl]$) location of $p_l^\star(i,:)$ set the same random number. The splitting procedure then the effect on $p_l^\star(i,:)$, which splits it into two random $1 \times (m+tl)$ row vectors $p_l'(i,:)$ and $p_l''(i,:)$. s_2 is a splitting indicator. If $s_2(j)$ is equal to 0, $p_l'(i,j)$ and $p_l''(i,j)$ are equal to $p_l^\star(i,j)$; if $s_2(j)$ is equal to 1, $p_l'(i,j)$ and $p_l''(i,j)$ are set as random values while their sum is equal to $p_l^\star(i,j)$. We obtain a resource location matrix encrypted as $I_L(i,:) = [p_l'(i,:) * LM_1^T, p_l''(i,:) * LM_2^T]$. After encrypting all the files, the data owner obtains $I_L = [p_l' * LM_1^T, p_l'' * LM_2^T]$.

Therefore, the encrypted index built is $I = I_C + I_L$. Finally, the data owner uploads C and I to the cloud.

By building an index of files, we obtain accurate information about them, so that we can provide users with accurate personalized search results.

C. BuildUM

We divide the user model into two parts: the user interest model and the user location model. For a detailed build step of the user interest model, see IV-C.2 and use the user's query to update the user interest model.

For a detailed build step of the user location model see IV-C.1. As Algorithm 1, PRMS obtains the GPS information (GPS_{ux}, GPS_{uy}) of a user, then calculates the distance between the user and the *location* as follows:

$$d(i) = \frac{distance(GPS_{ux}, GPS_{uy}, GPS_{ix}, GPS_{iy})}{180 * pi * 6370} \quad (5)$$

We then obtain the weight of each *location* $q_l(i) = \frac{1}{d(i)^2}$.

D. GenQuery (q, U, SK)

This has two steps:

Step 1 (Query Transformation): The system will generate a content query matrix based on the query keywords submitted by the user. The content query matrix changes according to the user interest model. When the weight of the keyword in the interest model is not 0, the weight of the keyword corresponding to the content query matrix becomes the weight of the user interest model. When the keyword weight in the user interest model is 0, the corresponding keyword weight in the query content matrix is unchanged, and is set as the initial value 1. So that the converted content query matrix not only contains the user's query information, but also contains the user's interest information. Meanwhile, by reading the location information in the user mobile device, the system generates the corresponding user location model as Algorithm 1 and equals the transformed location query matrix. Therefore the location query matrix contains the user's location information.

Algorithm 2 Encrypted Content Query Matrix

Input: $q; s_1; M_1^{-1}; M_2^{-1}$.
Output: Q_C.

```
tcqq=rand(1,tc−1)−0.5;
tcq=[tcqq −sum(tcqq(:))];
a=rand(1,1);
i = n + tc;
q_c=a*[q tcq];
r=rand(1,i);
for j = 1 : i do
  if s_1(j) == 1 then
    q'(j) = q_c(j);
    q''(j) = q_c(j);
  else
    q'(j) = r(j);
    q''(j) = q_c(j) − q'(j);
  end if
end for
Q_C = [q'*M1^{-1},q''*M_2^{-1}];
return Q_C
```

Step 2 (Query Encryption): The query encryption includes the encrypted content query matrix and encrypted user location query matrix. The encrypted content query matrix is described in Algorithm 2.

First, query q is extended to $(n+tc)$ dimensions as q_c. For the location from $(n+1)$ to $(n+tc-1)$, by randomly setting its value, the last dimension $(n+tc)$ is $-sum(tcqq(:))$. All locations are then scaled by a confused random number $a(a \neq 0)$.

After applying the splitting procedure as Algorithm 2, we calculate $Q_C = [q'M_1^{-1}, q''M_2^{-1}]$.

The encrypted user location matrix is similar to the encrypted query content matrix.

First, the user location matrix is extended to $(m+tl)$ dimensions as $q_l^\star$. For the location from $(m+1)$ to $(m+tl-1)$, by randomly setting its value, the last dimension $(m+tl)$ is -$sum(tlqq(:))$. Then, all locations are scaled by a confused random number $b(b \neq 0)$. $s_2(j)$ is the split

indicator. After applying the similar splitting procedure as above, the $Q_L = [q_l' LM_1^{-1}, q_l'' LM_2^{-1}]$. Therefore, $T = Q_C + Q_L = [q' M_1^{-1}, q'' M_2^{-1}] + [q_l' LM_1^{-1}, q_l'' LM_2^{-1}]$. The user sends the trapdoor and parameter K to the cloud.

E. RANKING(I, T, K)

In our PRMS scheme, users obtain the most relevant first K encrypted files. The cloud server computes the $F_g(i,:)$ of each file as equation 9, and returns the first K ranked as $file(1:K)$ to the user. The details of the method are described in Algorithm 3.

Algorithm 3 Ranking

Input: $I_C; Q_C; I_L; Q_L; K$.
Output: $file(1:K)$.
1: **for** $i = 1 : n$ **do**
2: $F_g(i,:)$=dot($I_C(i,:)$,Q_C)*dot($I_L(i,:)$,Q_L);
3: **end for**
4: $[\sim, file] = sort((F_g)^T, 'descend')$;
5: **return** $file(1:K)$;

The formula 6 defines the degree to which user queries and interests relate to each file:

$$dot(I_C(i,:), Q_C) = I_C(i,:) \cdot Q_C \tag{6}$$

where I_C is an encrypted, resource content matrix, and Q_C is an encrypted content query matrix.

Formula 7 defines the degree to which the location of the user relates to each file:

$$dot(I_L(i,:), Q_L) = I_L(i,:) \cdot Q_L \tag{7}$$

where I_L is the encrypted resource location matrix and Q_L is the encrypted user location matrix.

We use the law of gravitation to calculate the relevance of the score between the file and the user's search. The law of gravity is as follows:

$$F = \frac{GMm}{r^2} \tag{8}$$

We learn from the law of universal gravitation and rank the search results according to the following formula:

$$\begin{aligned} F_g(i,:) &= dot(I_C(i,:), Q_C) * dot(I_L(i,:), Q_L) \\ &= I_C(i,:) \cdot Q_C * I_L(i,:) \cdot Q_L \\ &= ab \frac{p(i,:) \cdot q * R(i)}{d(i)^2} \end{aligned} \tag{9}$$

where a, b are confused random numbers, $p(i,:)$ is a file vector, q is a query content vector, $R(i)$ is the characteristic of a *location*, and $d(i)$ is the distance between a user and a *location*.

The cloud server returns the files for the query. Higher values of F_g mean that the file better matches the user query, and the file should be at the front of the search results for one query issued by the user.

F. ANALYSIS

The importance of location information is obvious for location-based mobile search. The PRMS must rank the search results according to the users' profiles and location information. We learn from the law of universal gravitation, personalizing ranking according to the gravitational force between the trapdoor and index so that the user obtains highly relevant results from the cloud server.

VI. THE EFFICIENCY IMPROVEMENT OF PRMS

In order to improve the efficiency of the index construction and thus meet large data requirements, we propose an improved encryption algorithm in **BuildIndex(F, SK)**. Given that the content index and the location index are very similar to the construction, we have only described how to build the content index, which is shown in Algorithm 4.

Algorithm 4 Content Index

Input: $p; s_1; m_f; M_1^T; M_2^T$.
Output: I_C.
1: $h = n + tc$; where tc is the dimension of dummy keywords.
2: tcp=rand(1,1)*ones(1,tc);
3: **for** $i = 1 : m_f$ **do**
4: $p(i,:)$=[$p(i,:)$ tcp];
5: r=rand(1,h);
6: **for** $j = 1 : h$ **do**
7: **if** $s_1(j) == 1$ **then**
8: $p'(i,j) = r(1,j)$;
9: $p''(i,j) = p(i,j) - p'(i,j)$;
10: **else**
11: $p'(i,j) = p(i,j)$;
12: $p''(i,j) = p(i,j)$;
13: **end if**
14: **end for**
15: **end for**
16: $I_C = [p' * M_1^T, p'' * M_2^T]$;
17: **return** I_C

VII. SECURITY ANALYSIS

In this section, we analyze the PRMS scheme against the cloud server, which is honest and curious. Specific analysis is as follows:

Challenge: The cloud server manages all the ciphertext, encrypted index and the user's query history, and hopes to dig out some private information of the user from these data to learn interest and location. At the same time, the cloud server also wants to know the plaintext or index information of the file. If the cloud server knows the user's interests, location, or file plaintext or index, then the cloud server will win this game.

Theorem 1: The scheme can resist the first-level of cloud server attacks.

Proof: For the first-level attack, the cloud server only knows the ciphertext C, the index after encryption I, the encrypted query matrix T, and the encryption and decryption algorithm, which has no key and does not have enough information to crack the encrypted plaintext, index, and query matrix. The cloud server cannot know the user's interest or location; it is impossible to decrypt the file ciphertext into plaintext.

Theorem 2: The scheme can resist the second-level of cloud server attacks.

Proof: For the second-level attack, the cloud server knows the ciphertext, and the encryption and decryption algorithm, but does not have the key, so it is impossible to decrypt the ciphertext into the plaintext. However, the cloud server knows a part of the plaintext index p_A, so it may speculate on the privacy information of the user according to the gravitational force of the user query and each file index. By introducing the random number a and b (even if the query matrix q and q_l are the same) for the same file, the gravitational force between the user queries and the index is never the same. The proof of this is as follows:

$$\begin{aligned} F_{gc1}(i,:) &= dot(I_C(i,:), Q_{C1}) \\ &= I_C(i,:) \cdot Q_{C1} \\ &= a_1 p(i,:) \cdot q_1 \qquad (10) \\ F_{gc2}(i,:) &= dot(I_C(i,:), Q_{C2}) \\ &= I_C(i,:) \cdot Q_{C2} \\ &= a_2 p(i,:) \cdot q_2 \qquad (11) \\ a_1 \neq a_2, \quad q_1 &= q_2 \qquad (12) \end{aligned}$$

Available from equations 10, 11, and 12, $F_{gc1}(i,:) \neq F_{gc2}(i,:)$. Therefore, it is impossible for the cloud server to deduce privacy information such as the user's interest.

$$\begin{aligned} F_{gl1}(i,:) &= dot(I_L(i,:), Q_{L1}) \\ &= I_L(i,:) \cdot Q_L \\ &= b_1 \frac{R(i)}{d_1(i)^2} \qquad (13) \\ F_{gl2}(i,:) &= dot(I_L(i,:), Q_{L2}) \\ &= I_L(i,:) \cdot Q_L \\ &= b_2 \frac{R(i)}{d_2(i)^2} \qquad (14) \\ b_1 \neq b_2, \quad d_1(i) &= d_2(i) \qquad (15) \end{aligned}$$

Available from equations 13, 14, and 15, $F_{gl1}(i,:) \neq F_{gl2}(i,:)$. Therefore, it is impossible for the cloud server to deduce privacy information such as the location of the user.

$$\begin{aligned} F_{g1}(i,:) &= F_{gc1}(i,:) * F_{gl1}(i,:) \\ &= a_1 b_1 \frac{p(i,:) \cdot q_1 * R(i)}{d_1(i)^2} \qquad (16) \\ F_{g2}(i,:) &= F_{gc2}(i,:) * F_{gl2}(i,:) \\ &= a_2 b_2 \frac{p(i,:) \cdot q_2 * R(i)}{d_2(i)^2} \qquad (17) \end{aligned}$$

As shown in Equation 10 - 17, each time the user randomly generates the value of a and b, the cloud server cannot deduce the user's private information. The cloud server returns the ciphertexts that are most relevant to the user's location, interests, and query keywords.

Theorem 3: The scheme can resist the third-level of cloud server attacks.

Proof: Compared with the second-level attack of the cloud server, in the third-level attack, the cloud server knows part of the plaintext index p_A and its corresponding ciphertext I_A, so that the cloud server may adopt a violent attack method. The scheme divides q, q_l, p, p_l into two matrices by introducing two split indicator row matrices s_1, s_2, and the cloud service must know s_1, s_2 if it is to be cracked. In order to increase the security of the scheme, the dimensions of q, q_l, p, p_l have been expanded, the dimensions of q, p will be extended into $n+tc$ dimensions, the dimensions of q_l, p_l will be extended into $m+tl$ dimensions. Because split indicators s_1, s_2 are binary vectors, there are two choices in each value. The cloud server needs to try $2^{n+m+tc+tl}$ times if it must be cracked. If we let $n+m+tc+tl = 100$, the cloud server tries 10^{12} times per second. It also requires nearly 4×10^{10} years. Therefore, the scheme can resist the violent attack of the cloud server (that is, the scheme achieves the challenge of plaintext attack security and ensures that the privacy of the user and the index is not leaked).

VIII. PERFORMANCE EVALUATION

We use the "business" and "review" data in the Yelp dataset to analyze this work. We randomly select different numbers of "business" data and "review" data to build the dataset.

The entire experiment is implemented on a 2.6GHz Intel (R) Core (TM) i7-6700HQ CPU, Windows 10 operating system with RAM of 16GB. We use Matlab R2016b to implement the simulation code, and originPro 2017 was used to simulate the experimental data.

A. PRECISION

Users in the search process, especially in mobile search scenarios, want to obtain the search results that best match the search intent. The PRMS scheme optimizes the query result through the query keywords submitted by the user, user interest, and the user's location to ensure the accuracy of the user query. To prove the accuracy of the method, we randomly selected 10 users. As expected, 9 of them were satisfied with our return results, which also shows the precision of our scheme. The cloud server will return the first K files to the user based on the parameters K. If $K = 10$, then there are 100 files related to the query, and our recall rate will be very low. Therefore, it is meaningless to discuss the recall rate for our scheme.

B. INDEX CONSTRUCTION

In order to meet the requirements of mobile searches, in the PRMS scheme, the index construction is divided into the content index construction and the location index construction. We compare two schemes to built the index using matrix encryption: MRSE [8] and PRSE [3].

1) CONTENT INDEX CONSTRUCTION

To construct the content index, we first calculate the weight of keywords. We randomly select different numbers of "business" data and "review" data in Yelp to build the dataset. In order to reflect a "business" in more detail, we combine the "business" information and "reviews" about this "business" as final "business" information.

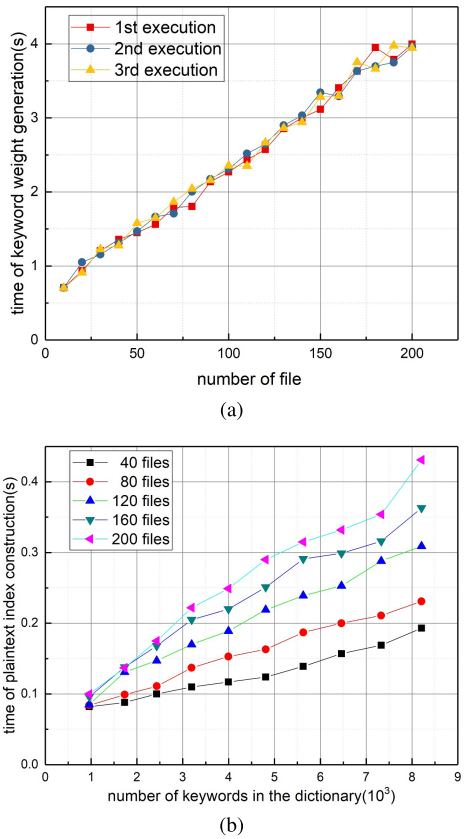

FIGURE 4. Time of file vector construction: (a) User TF-IDF to generate the weight of the keyword for the different size of file collection; (b) For the different size of keyword dictionary with the different size of file collection.

We compute the "$TF \times IDF$" of each keyword. Figure 4(a) shows the time of keyword weight generation increases with the number of files. Next, we construct a plaintext index $p(i, :)$, using Spyder (Python 2.7) to implement the simulation code written in Python. For the same size of the file collection, Figure 4(b) shows the time of plaintext index construction increases with the number of keywords.

In PRMS, the matrix is extending and splitting. The matrix is encrypted as in Algorithm 4, so as to obtain an encrypted content index matrix. Figure 5(a) shows that the time cost of content index construction increases with the number of files. As can also be seen from Figure 5(a), when the number of keywords in the dictionary is 18711 and the number of files increases from 2000 to 10000, the construction time of the PRMS_improvement increases from 140s to 2505s, the PRMS increases from 647s to 6016s, the MRSE increases from 6211s to 32243s, and the PRSE increases

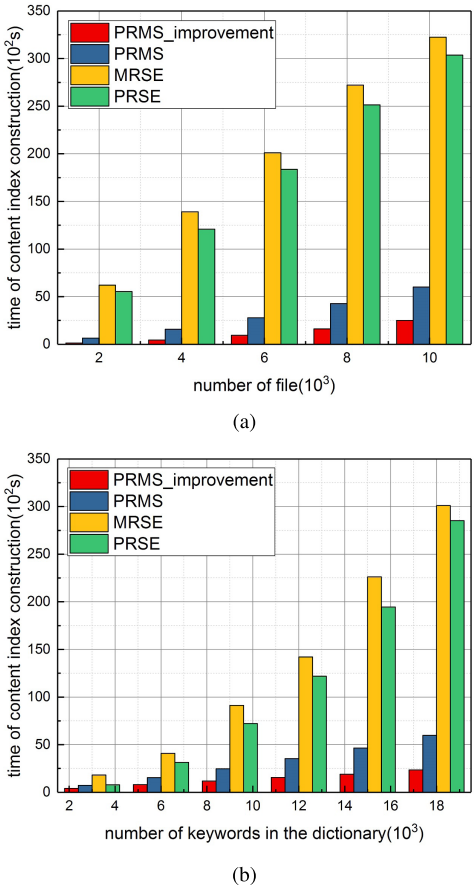

FIGURE 5. Time of content index construction: (a) For the different size of file collection with the same keyword dictionary, $n + tc = 18,711$; (b) For the different size of keyword dictionary with the same file collection, $m_f = 10,000$.

from 5531s to 30360s. We conclude that PRMS_improvement scheme is far more efficient than MRSE [8] and PRSE [3].

Figure 5(b) shows that the time cost of content index construction (including extending, splitting, and encrypting produce) increases with the number of keywords in the dictionary. From this, we also see that when the number of files is 10000, the number of keywords in the dictionary increased from 3000 to 18000, index construction time of the PRMS_improvement increases from 409s to 2340s, the PRMS increases from 722s to 5983s, the MRSE increases from 1812s to 30111s, and the PRSE increases from 789s to 28517s. In addition, the time cost of content index construction of the PRMS_improvement and the PRMS is nearly linear with the number of keywords, but the MRSE and the PRSE showed a quadratic parabola with the number of keywords. Therefore, the larger the number of keywords in the dictionary, the more obvious the advantages of the PRSE.

As can be seen from Algorithm 4, the PRMS_improvement aggregates all the file vectors into a matrix, and encrypts the matrix after extending and splitting. Therefore, encryption only needs to be done once regardless of the number of files.

The PRMS_improvement greatly reduces the time to build the index, reducing the burden of the data owner, so the

TABLE 1. Storage of subindex/query.

Size of keyword dictionary	3000	6000	9000	12000	15000	18000
MRSE(KB) [8]	23.44	46.88	70.31	93.75	117.19	140.63
PRSE(KB) [3]	46.88	93.75	140.63	187.5	234.37	281.25
PRMS(KB)	48	96	144	192	240	288

TABLE 2. Storage of user interest/location model.

Size of query terms	500	1000	1500	2000	2500	3000
PRSE(KB) [3]	12.20	23.43	38.08	50.78	63.47	76.17
PRMS(KB)	4	8	12	16	20	24

choose of the PRMS_improvement to build the index is a good choice.

2) LOCATION INDEX CONSTRUCTION

In order to give the user location-based search results, we build a location index $I_L(i,:)$ for each file F_i in the dataset. The first step is to obtain the *business* name as the keyword and *stars* as the keyword weight to generate a $p_l(i,:)$. Next, the matrix is extended and split. Finally, the matrix is encrypted. In our PRMS scheme, the method of extending, splitting, and encryption is the same as content index construction, therefore our PRMS scheme is more efficient than MRSE [8] and PRSE [3].

We analyzed the storage overhead of each content subindex, and each location subindex and query in Table 1 with different sizes of the keyword dictionary compared to MRSE and PRSE. The storage of each subindex/query is linearly related to the size of the keyword dictionary. The storage of PRMS and PRSE is almost the same. However, they are twice the MRSE, due to the data structure of MRSE is ''int'', but, PRMS and PRSE are ''double'', which expresses the keyword weight.

C. USER MODEL CONSTRUCTION

The user model implies the user's personal information, including the interests and location.

1) USER INTEREST MODEL CONSTRUCTION

In order to return the ciphertext that is in accordance with the user's interest, this paper uses the method of article [3] to build a user interest model, which uses semantic information through user's query history and WordNet [40]. Figure 6(a) shows that the update time of the user interest model increases with the number of keywords, but has nothing to do with the number of keywords in the query.

We analyzed the storage overhead of the user model in Table 2 with different query sizes compared to PRSE, which is linearly related to the query size. The storage of PRMS is significantly lower than PRSE.

2) USER LOCATION MODEL CONSTRUCTION

As mentioned above, the user location model is related to the location when the user queries (that is, it is related to the GPS information from the mobile device when the user queries). PRMS obtains the GPS information as the user location information when the user queries. The system computes distance between the user location and ''business'' *location* in the database to generate the user location model. Due to the same complexity of computing the distance, the time cost is the same even if the *location* is different, as shown in Figure 6(b).

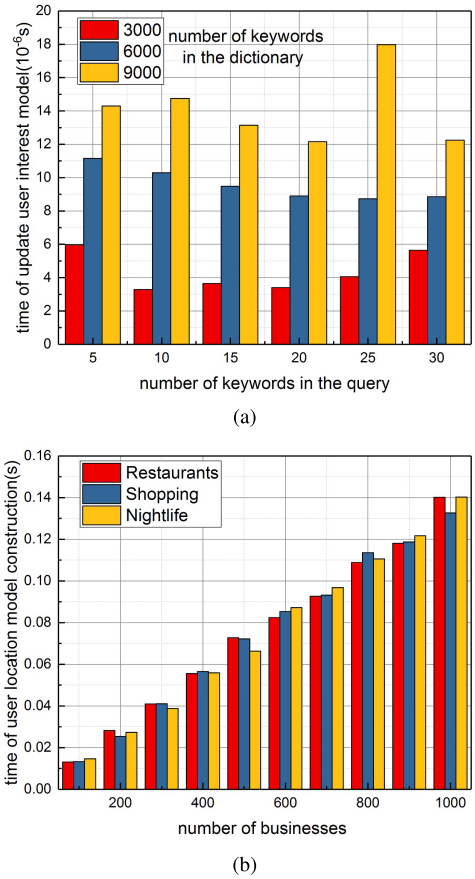

FIGURE 6. User model construction: (a) Update of user interest model; (b) Time of user location model construction.

D. QUERY GENERATION

In our paper, a query consists of two parts: content query and location query. Figure 7 shows that the time of location query generation is greater than the time of content query generation when the number of keywords is the same. This is because the user location query includes the user location model, which must calculate the distance between the user and each *location*, as is shown in Algorithm 1.

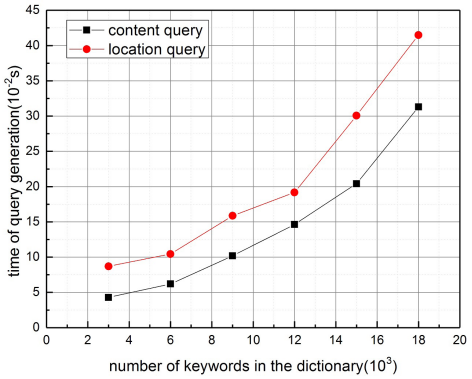

FIGURE 7. Time of query generation.

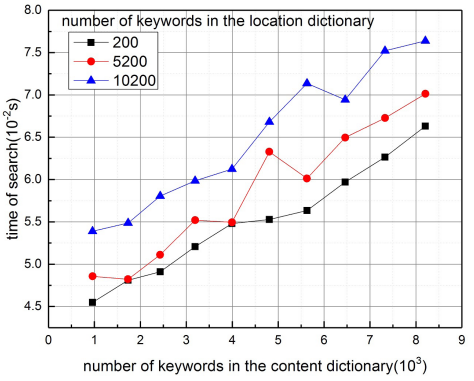

FIGURE 8. Time of search.

E. SEARCH

Search performance includes computing and ranking gravitational forces for all files in the dataset. When the size of the file in the dataset is equal to 200, Figure 8 shows that the search time is dominated by the number of keywords in the content dictionary and the location dictionary.

TABLE 3. Compare personalized search results.

Top 10 files
Consider location information
32 154 181 186 34 160 66 67 46 90
Regardless of location information
186 32 154 160 66 90 46 96 119 136

When the size of the file in the dataset is 200, for a user query, Table 3 shows the top 10 files returned when location information is considered or ignored. There are 3 different files returned from the cloud server in two conditions. 181, 34, 67 are returned when considering location information, and 96, 119, 136 are returned when disregarding location information. It is enough to illustrate the importance of location information in personalized searches. However, there are 7 identical files (32, 154, 186, 160, 66, 46, 90) returned from the cloud server in two conditions, indicating that the user query and the user interest model are important in the personalized search.

IX. CONCLUSION

In the mobile environment, we proposed a personalized mobile search over encrypted outsourced data to solve the problem of information overload and privacy protection. We first proposed an encryption method and applied it to construct the content index and location index, which improved our PRMS scheme. Through the PRMS scheme, we realized personalized mobile searches based on user's interests and location information. We used the law of gravitation to sort the search results in order to obtain better personalized search results. Through security analysis, performance evaluation, and experiments on the Yelp dataset, our proposed scheme is shown to be practical.

REFERENCES

[1] S. Zhang, K.-K. R. Choo, Q. Liu, and G. Wang, "Enhancing privacy through uniform grid and caching in location-based services," *Future Gener. Comput. Syst.*, to be published, doi: 10.1016/j.future.2017.06.022.
[2] T. Peng, Q. Liu, D. Meng, and G. Wang, "Collaborative trajectory privacy preserving scheme in location-based services," *Inf. Sci.*, vol. 387, pp. 165–179, May 2017.
[3] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.
[4] E. Agichtein, E. Brill, S. Dumais, and R. Ragno, "Learning user interaction models for predicting Web search result preferences," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2006, pp. 3–10.
[5] Q. Tan, X. Chai, W. Ng, and D.-L. Lee, "Applying co-training to clickthrough data for search engine adaptation," in *Database Systems for Advanced Applications* (Lecture Notes in Computer Science), vol. 2973. Berlin, Germany: Springer, 2004, pp. 519–532.
[6] K. W.-T. Leung, D. L. Lee, and W.-C. Lee, "PMSE: A personalized mobile search engine," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 820–834, Apr. 2013.
[7] Q. Du *et al.*, "Folksonomy-based personalized search by hybrid user profiles in multiple levels," *Neurocomputing*, vol. 204, pp. 142–152, Sep. 2016.
[8] 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.
[9] X. Jiang, J. Yu, J. Yan, and R. Hao, "Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data," *Inf. Sci.*, vols. 403–404, pp. 22–41, Sep. 2017.
[10] 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.
[11] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2005, pp. 442–455.
[12] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.
[13] Z. Liu, T. Li, P. Li, C. Jia, and J. Li, "Verifiable searchable encryption with aggregate keys for data sharing system," *Future Generat. Comput. Syst.*, vol. 78, pp. 778–788, 2018.
[14] 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.
[15] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 3027. Berlin, Germany: Springer-Verlag, 2004, pp. 506–522.
[16] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2005, pp. 414–426.

[17] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and Network Security* (Lecture Notes in Computer Science), vol. 3089. Berlin, Germany: Springer, 2004, pp. 31–45.
[18] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. Int. Conf. Pairing-Based Cryptogr.*, 2007, pp. 2–22.
[19] 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 *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 2013.
[20] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 72–75.
[21] Z. Liu, J. Weng, J. Li, J. Yang, C. Fu, and C. Jia, "Cloud-based electronic health record system supporting fuzzy keyword search," *Soft Comput.*, vol. 20, no. 8, pp. 3243–3255, 2016.
[22] M. Speretta and S. Gauch, "Personalized search based on user search histories," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, Sep. 2005, pp. 622–628.
[23] Y.-C. Yu, "A social interaction system based on cloud computing for mobile video telephony," *Sci. China Inf. Sci.*, vol. 57, no. 3, pp. 1–10, 2014.
[24] M. Leginus, C. Zhai, and P. Dolog, "Personalized generation of word clouds from tweets," *J. Assoc. Inf. Sci. Technol.*, vol. 67, no. 5, pp. 1021–1032, 2016.
[25] L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li, "Personalized recommendation via parameter-free contextual bandits," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2015, pp. 323–332.
[26] H. Wang, X. He, M.-W. Chang, Y. Song, R. W. White, and W. Chu, "Personalized ranking model adaptation for Web search," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2013, pp. 323–332.
[27] H. Xie *et al.*, "Personalized search for social media via dominating verbal context," *Neurocomputing*, vol. 172, pp. 27–37, Jan. 2016.
[28] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," in *Proc. Int. Conf. Distrib. Comput. Syst. Workshops*, 2011, pp. 273–281.
[29] C. Liu, L. Zhu, L. Li, and Y. Tan, "Fuzzy keyword search on encrypted cloud storage data with small index," in *Proc. IEEE Int. Conf. Cloud Comput. Intell. Syst.*, Sep. 2011, pp. 269–273.
[30] Q. Zhang, Q. Liu, and G. Wang, "A privacy-preserving hybrid cooperative searching scheme over outsourced cloud data," in *Proc. Int. Conf. Secur., Privacy Anonymity Comput., Commun. Storage*, 2016, pp. 265–278.
[31] H. Li, D. Liu, Y. Dai, T. H. Luan, and S. Yu, "Personalized search over encrypted data with efficient and secure updates in mobile clouds," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 97–109, Jan./Mar. 2018.
[32] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Cooperative private searching in clouds," *J. Parallel Distrib. Comput.*, vol. 72, no. 8, pp. 1019–1031, 2012.
[33] E. Luo, Q. Liu, J. H. Abawajy, and G. Wang, "Privacy-preserving multi-hop profile-matching protocol for proximity mobile social networks," *Future Gener. Comput. Syst.*, vol. 68, pp. 222–233, Mar. 2017.
[34] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. Conf. Inf. Commun.*, 2010, vol. 62, no. 2, pp. 525–533.
[35] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu, "Exploring folksonomy for personalized search," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, Singapore, Jul. 2008, pp. 155–162.
[36] O. Alhabashneh, R. Iqbal, F. Doctor, and A. James, "Fuzzy rule based profiling approach for enterprise information seeking and retrieval," *Inf. Sci.*, vols. 394–395, pp. 18–37, Jul. 2017.
[37] D. Datta, S. Varma, C. C. Ravindranath, and S. K. Singh, "Multimodal retrieval using mutual information based textual query reformulation," *Expert Syst. Appl.*, vol. 68, pp. 81–92, Feb. 2017.
[38] Z. Fu, X. Wu, Q. Wang, and K. Ren, "Enabling central keyword-based semantic extension search over encrypted outsourced data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 12, pp. 2986–2997, Dec. 2017.
[39] S. Zhang, G. Wang, Q. Liu, and J. H. Abawajy, "A trajectory privacy-preserving scheme based on query exchange in mobile social networks," *Soft Comput.*, pp. 1–13, Jun. 2017, doi: 10.1007/s00500-017-2676-6.
[40] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

QIANG ZHANG received the B.Sc. degree from Central South University, Changsha, China, in 2011, and the M.Sc. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2014. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, School of Information Science and Engineering, Central South University. His research interests include privacy preserving, cloud computing, and information retrieval.

QIN LIU received the B.Sc. degree from Hunan Normal University, Changsha, Hunan, China, in 2004, and the M.Sc. and Ph.D. degrees from Central South University, Changsha, in 2007 and 2012, respectively, all in computer science. She has been a Visiting Student with Temple University, Philadelphia, PA, USA. Her research interests include security and privacy issues in cloud computing. She is currently an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha.

GUOJUN WANG (M'08) received the B.Sc. degree in geophysics and the M.Sc. and Ph.D. degrees in computer science Central South University, Changsha, China, in 1992, 1996, and 2002, respectively. He was a Professor with Central South University, China, an Adjunct Professor with Temple University, USA, a Visiting Scholar with Florida Atlantic University, USA, a Visiting Researcher with the University of Aizu, Japan, and a Research Fellow with The Hong Kong Polytechnic University. He is currently a Professor with the School of Computer Science and Educational Software, Guangzhou University. His research interests include network and information security, Internet of Things, and cloud computing. He is a Distinguished Member of CCF and a member of ACM and IEICE.