Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement

Zhangjie Fu, *Member, IEEE*, Kui Ren, *Fellow, IEEE*, Jiangang Shu, Xingming Sun, *Senior Member, IEEE*, and Fengxiao Huang

Abstract—In cloud computing, searchable encryption scheme over outsourced data is a hot research field. However, most existing works on encrypted search over outsourced cloud data follow the model of "one size fits all" and ignore personalized search intention. Moreover, most of them support only exact keyword search, which greatly affects data usability and user experience. So how to design a searchable encryption scheme that supports personalized search and improves user search experience remains a very challenging task. In this paper, for the first time, we study and solve the problem of personalized multi-keyword ranked search over encrypted data (PRSE) while preserving privacy in cloud computing. With the help of semantic ontology WordNet, we build a user interest model for individual user by analyzing the user's search history, and adopt a scoring mechanism to express user interest smartly. To address the limitations of the model of "one size fit all" and keyword exact search, we propose two PRSE schemes for different search intentions. Extensive experiments on real-world dataset validate our analysis and show that our proposed solution is very efficient and effective.

Index Terms—Cloud security, outsourcing security, personalized search, user interest model

1 Introduction

In recent years, cloud computing has achieved great development both in academic and industry communities as it provides economic and convenient service. And now more and more companies and users are planning to upload their data onto the public clouds. However, data stored in the cloud may suffer from malicious use by cloud service providers since data owners have no longer direct control over data. Considering data privacy and security, it is a recommended practice for data owners to encrypt data before uploading onto the cloud. Although it protects data security from illegal use both from untrusted cloud service providers and external users, it makes data utilization more difficult since many techniques based on plaintext are no longer applicable to ciphertext. Therefore, exploring an efficient search technique for encrypted data is extremely urgent.

A popular way to search over encrypted data is searchable encryption [1] and many constructive schemes have been put forward under different applications. However, these searchable encryption schemes based on keyword no longer fully satisfy the new challenge and users' increasing needs, specifically manifested in the following two aspects.

One is that most of existing schemes follow the model of "one size fits all" and ignore individual users' experience due to their different hobbies, interests or cultural backgrounds. In those schemes, the cloud will return all files that match the user's query, which may cause a huge consumption of network bandwidth. Moreover, it will cost user much time and many resources to filter his real interesting ones among a large quantity of returned files. In the practical application, *different users may find different things relevant because of different importance or priorities of query terms*, indicating the necessity of *personalized search*, which *takes personal keyword preference or keyword priority into account*. So how to design an efficient search scheme that can really understand the user's search intention is a pressing problem. In general, personalized search [26] in the field of Information Retrieval (IR) may be a good choice, which can extract user information to optimize the ranking of result. However, these schemes cannot be directly applied in searchable encryption schemes due to the lack of consideration of privacy and security. Shen et al. [19] proposed a preferred keyword search scheme over encrypted data, but the artificial manner of measuring keyword preference has great randomness and fails to consider different users' search histories.

The other one is that most of these schemes support only exact keyword search. That means the returned result is only related to the user's input. When the user queries some uncommon terms, it is possible that just a few matched results are returned and the user may be not satisfied with the returned results. Besides, most users are typically untrained and casual, they might not input the explicit query terms that accurately match their true search intentions. As a result, the returned result is definitely not what users really want. In such cases, users want to retrieve more

- *Z. Fu is with the School of Computer and Software, Jiangsu Engineering Centre of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, China and the Department of Computer Science and Engineering, The State University of New York at Buffalo, Buffalo, NY 14260. E-mail: zhangjie@buffalo.edu.*
- *K. Ren is with the Department of Computer Science and Engineering, The State University of New York at Buffalo, Buffalo, NY 14260. E-mail: kuiren@buffalo.edu.*
- *J. Shu, X. Sun, and F. Huang are with the School of Computer and Software, Jiangsu Engineering Centre of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, China. E-mail: kennethshu@126.com, {sunnudt, hfx20101344034}@163.com.*

Manuscript received 26 July 2015; revised 23 Oct. 2015; accepted 28 Nov. 2015. Date of publication 8 Dec. 2015; date of current version 10 Aug. 2016. Recommended for acceptance by X. Wang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2506573

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.
1045-9219 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

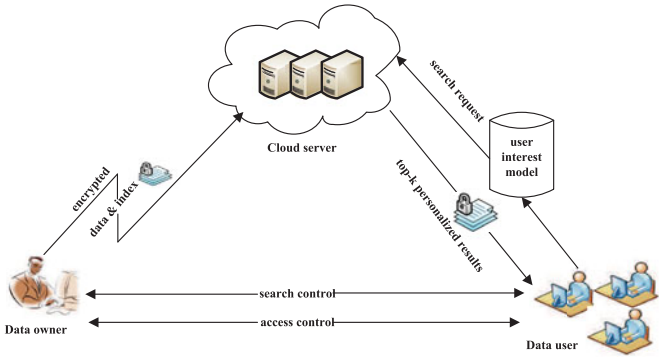


Fig. 1. Architecture of the search over encrypted cloud data.

results as similar as possible to query terms. To solve this problem, the common technique used in IR is query extension, which can extend original query terms according to some rules before submitting search request. Similarly, query extension remains to be adjusted and improved to meet searchable encryption in cloud computing.

Therefore, how to design a searchable encryption scheme with support of both personalized ranking and query extension, is the problem that we try to tackle in this paper. In this paper, for the first time, we study and solve the problem of personalized multi-keyword ranked search over encrypted data (PRSE) while preserving privacy in the cloud computing. In PRSE, with the help of semantic ontology WordNet, user interest model for individual user is built by analyzing the user's search history. And we adopt a scoring mechanism to express user interest smartly by calculating the similarity score between different types of related words and the keyword. We propose a basic design of PRSE, and then give two PRSE schemes based on secure inner product [14] in order to meet different search intentions.

The new contributions in this paper compared with [11], [12], [13], [14], [15], [16], [19], [32], [33] are summarized as follows:

1) For the first time, we explore the problem of personalized multi-keyword ranked search over encrypted cloud data. Compared with [19], we build a user interest model to record and analyze the user's search history, score the keyword smartly as keyword priority using personal keyword access frequency.
2) With the help of user interest model built upon Wordnet, we can also support query semantic extension to help the user to get more related files.
3) To address the limitations of the model of "one size fit all" and keyword exact search, we propose two PRSE schemes for different search intentions.
4) Through rigorous analysis and experiments on the real-world dataset, our proposed schemes are efficient and feasible.

The rest of this paper is organized as follows. Section 2 presents the system model, threat model and our design goals and then briefly describes some notations and background knowledge used in our paper. Section 3 depicts the basic design of PRSE, followed by Section 4, which shows two proposed schemes in detail. We improve search efficiency of PRSE in Section 5. Sections 6 and 7 presents security analysis and performance evaluation, respectively. We summarize related work on searchable encryption and personalized search in Section 8 and finally conclude the paper in Section 9.

2 Problem Formulation

2.1 System Model

A complete system model in cloud computing should involve three different entities: the data owner, the data user and the cloud server, as shown in Fig. 1. Different from the previous work [11], [12], [13], [14], [15], [16], there exists a user interest model stored in the user side. The user interest model (see Section 3.3) is built upon the user's long-term search history. It records access frequency of both query keywords and their related keywords with the help of WordNet. Different access frequency of keywords, as keyword priority (see Section 2.5), can reflect their different importance in viewpoint of the data user. To search for files of interest, the data user should firstly produce a search request. And then query reformulation that achieves keyword priority of query terms will be carried out through the user interest model. At last, the encrypted search query through search control mechanism, e.g., broadcast encryption [4], will be sent to the cloud. Upon receiving the search request from the authorized user, the cloud server will conduct designated search operation over the index and send back relevant encrypted documents, which have been well ranked by the cloud server according to some ranking criteria (e.g., relevance score in Section 2.5).

2.2 Notations

- D—the plaintext document collection, denoted as $D = (D_1, D_2, \ldots, D_m)$.
- C—the encrypted document collection, denoted as $C = (C_1, C_2, \ldots, C_m)$.
- W—the keyword set consisting of n keywords, denoted as $W = (w_1, w_2, \ldots, w_n)$.
- p_i—the index vector for document D_i, denoted as $p_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,n})$ where each dimension is a normalized keyword weight.
- P—the plain index for D, denoted as $P = \{p_1, p_2, \ldots, p_m\}$.
- P_A—a set of plain index, $P_A \subseteq P$.
- I—the encrypted searchable index for each document, denoted as $I = (I_1, I_2, \ldots, I_m)$ where each index I_i is based on p_i.
- I_A—the corresponding encrypted values of P_A, $I_A \subseteq I$.
- $\widetilde{W}$—the plaintext query words containing t keywords, the subset of W, denoted as $\widetilde{W} = (w_{j_1}, w_{j_2}, \ldots, w_{j_t})$.
- q—the query vector, denoted as $q = (q_1, q_2, \ldots, q_n)$ where each dimension is a keyword priority.
- T—the trapdoor of the query, which is based on q.
- $R_{T,k}$—the personalized ranked id of top-k documents according to the relevance score with T.
- U—the user interest model built on the search history.
- $p_{i,z}$—the zth keyword weight in the index vector p_i.

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

2.3 Threat Model

Referring to previous work [14], the cloud server is considered as honest-but-curious. Specifically, the cloud server is expected to execute designated protocol honestly and correctly, but it is eager to get some sensitive information by inferring and analyzing data or index in its storage and the search request received during the protocol. In our paper, we assume that besides ciphertext, the cloud server also knows what encryption and decryption algorithm are being used. However, he doesn't know the key. We assume the objective of the cloud server is to recover some plain index $P_A \subseteq P$. Therefore, our objective is to prohibit the cloud server from obtaining P_A or P. Apart from the ciphertext, the cloud server may possess additional knowledge about the original data. To better evaluate the security of a scheme, we classify cloud servers into different levels based on the knowledge H they have.

- Level 1: the cloud server observes only the ciphertext, i.e., $H =< C, I, T >$.
- Level 2: besides the ciphertext, the cloud server also knows a set of plain index P_A, i.e., $H = < C, I, T, P_A >$.
- Level 3: the cloud server observes a set of plain index P_A and he also knows those corresponding encrypted values I_A, i.e., $H =< C, I, T, P_A, I_A >$.

2.4 Design Goals

- *Personalized ranked search*: PRSE should support personalized ranking of results considering different users' interests, such as keyword priority or preference.
- *Query semantic extension*: Based on the deep understanding of the user's search intension, our scheme should support query extension before submitting so as to solve the limitation of keyword exact search.
- *Privacy-preserving*: The general goal is to prevent the cloud server from learning additional sensitive information from document collection, index and search request. Specifically, we are concerned with privacy requirements: Keyword Privacy, Index Confidentiality, Query Confidentiality, and Trapdoor Unlinkability. We do not aim to protect access pattern (sequence of search results) in consideration of efficiency [21].

2.5 Preliminaries

We now introduce some necessary background knowledge for our proposed scheme:

Keyword weight. Keywords are practical tools to summarize document content. In order to express keyword's significance to the document, we adopt the most widely statistical measurement "TF×IDF", where term frequency (TF) is the occurrence of the term appearing in the document, and inverse document frequency (IDF) is usually obtained by dividing the total number of document collection by the number of documents containing the term. Specially, TF represents the importance of the term within a document and IDF indicates the importance or degree of distinction within the whole document collection. Here we calculate the keyword weight with the formula below:

$$S_{t,D_d} = \frac{\ln(1 + f_{d,t}) \times \ln(1 + \frac{m}{f_t})}{\sqrt[2]{\sum_{t \in D_d} (\ln(1 + f_{d,t}) \times \ln(1 + \frac{m}{f_t}))^2}}. \quad (1)$$

Where $f_{d,t}$ is the TF of the term t in document D_d, f_t is the number of documents containing the term t and m is the number of the whole document collection. The numerator is the value of the most typical variation of "TF×IDF" and the denominator just functions as the normalization factor. Through the formula, we can get the keyword weight of keyword in a document. Comparing with the scheme presented in [14] which ignores keyword weight, our model is more practical and scientific.

Keyword priority. Through the well-trained user interest model, we can get the access frequency of each keyword. The higher the access frequency of a keyword is, the more important the keyword is from viewpoint of the user. The importance of the keyword usually means its rank priority.

Relevance score. It is used to measure the score of the query to a document. We can divide the whole relevance score into many sub-scores to represent the connection of file to the keywords in the query. The product of the keyword weight and the keyword priority is regarded as the sub-score, and thus the accumulated sub-scores form the relevance score of the query to a document. We can express the keyword weight and the keyword priority as a vector, and so the inner product of these two vectors can achieve the same effect.

Secure inner product. When calculating the relevance score of the document to the query, we should use two vectors: the document vector and the query vector. However, it is not advisable to directly outsource two vectors onto the cloud at the risk of leaking index privacy and query privacy. To achieve the goal of privacy-preserving, we adopt the secure inner product in [14]. The algorithm can make the cloud compute the inner product of two encrypted vectors $E(\overrightarrow{p})$ and $E(\overrightarrow{q})$ without knowing the actual values of $\overrightarrow{p}$ and $\overrightarrow{q}$ so that $E(\overrightarrow{p}) \cdot E(\overrightarrow{q}) = p \cdot q$. The details will be described later.

WordNet. WordNet [20] is a lexical database available online, which offers a large repository of English lexical items. It is organized hierarchically by a collection of synset, the smallest unit, which represents a specific meaning of a word. Synsets are connected to one another through some conceptual-semantic and lexical relations, including synonym, hypernym, hyponym, meronym, holonym and so on. These relations play an important role in computational linguistics and natural language processing. For brevity, only three kinds of relations: Synonymous relation, Hypernym/Hyponym relation, Meronym/Holonym relation are used in building of the user interest model.

3 The Basic Design

3.1 Overview

In PRSE, we adopt "secure inner product similarity" like MRSE [14] to quantitatively evaluate relevance score of the query to a document, as shown in Fig. 2. In our PRSE, each document is associated with a multidimensional vector and the corresponding dimension is a keyword weight if the

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

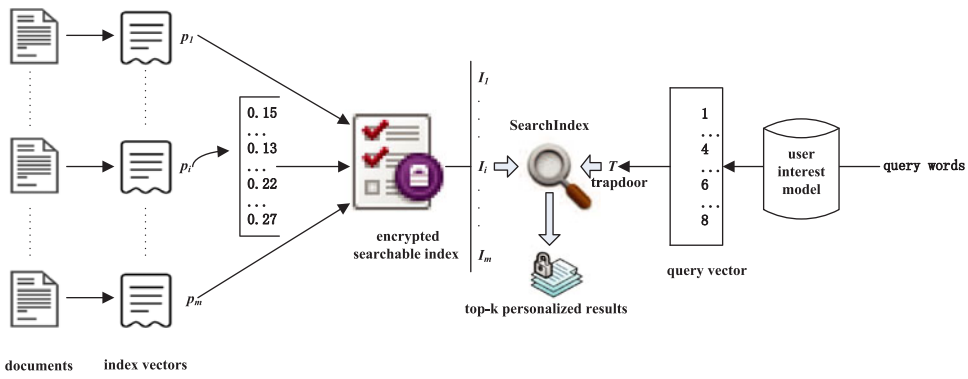


Fig. 2. Overview of PRSE scheme.

document contains the keyword. Similarly, the search query is also described as a multidimensional vector where each dimension is the corresponding keyword priority in the user interest model. The user interest model is trained from the user's long-term search history. Through our scheme, different users can get different results even for the same query words, realizing the goal of personalized ranking.

3.2 Framework

The PRSE system consists of the following algorithms:

- **Setup:** In this initialization phase, the data owner produces a symmetric key as SK.
- **BuildIndex**(D, SK): Based on the keyword set W extracted from the document collection D, the data owner can build a searchable index I, which is encrypted by the symmetric key SK, and encrypt the document collection independently. After that, the data owner outsources the index I and encrypted documents C onto the cloud.
- **BuildUIM:** The system collects the user's long-term search history, records the access frequency of query keywords, and then builds the user interest model.
- **GenQuery**$(\widetilde{W}, U, SK)$**:** With the query words $\widetilde{W}$, the data user conducts query reformulation with the user interest model and then produces a corresponding trapdoor T encrypted by the same symmetric key SK. At last, T, together with parameter k, will be submitted to the cloud.
- **SearchIndex**(I, T, k)**:** Upon receiving the trapdoor T, the cloud server conducts designated search operation over the index I, and returns $R_{T,k}$ sorted by the relevance score between documents and T.

3.3 User Interest Model

In order to realize the goals of personalized ranking and query semantic extension, we should build a personalized user interest model for each individual to recognize each other. The user interest model is built upon the user's long-term search history with the help of the semantic ontology WordNet. Here we adopt the incremental updating strategy to build the user interest model with scoring mechanism. In particular, the user interest model will be updated once a new keyword comes. For a certain keyword, we extract some related words which have semantic relations with the keyword and add them into the user interest model as nodes. And we give different scores of the related words according to their different types of relations.

Question: Why to give scores to related words?

Answer: The user interest model is based on personal keyword access frequency in essence. Assume that when a keyword (e.g., "desk") comes, we set the score of the keyword as 1. That means the keyword appears just once. While for some related words (e.g., "table", "secretaire") of the keyword, if they are set the score 0, it is unreasonable. The reason is that for the keyword "desk", "table" is its synonym and "secretaire" is a kind of "desk" (They all have some semantic relations with the keyword). Considering semantic relation, we give them different scores according to different types of relations with the keyword to represent their appearing times.

When a new keyword has already existed in the user interest model, we should multiply by an impact factor θ ($\theta \in (0, 1]$) when increasing the corresponding score. It means what percentage of the old score contributes to the new score. In fact, in order to guarantee the new score is larger than the old score during each updating, θ should be fixed as 1. As a result, the more frequent keywords appeared in the user interest model are, the larger their scores are. The details of the update is described in Algorithm 1. Note that α, β, γ in algorithm 1 are just tags for distinctions of three different relationships. They represent similarity score between the related word and the keyword, which can be calculated by WordNet.

If a user has a relatively fixed interest in a long period of time, considering the cost of updating the user interest model, the user interest model can be fixed after enough time to train. When the user changes his interest in future (e.g., the user's query words are often out of the user interest model), the user interest model will be rebuilt. And we can also choose the way to update the user interest model constantly to meet the user with slowly changing interest.

4 PRSE Schemes

In order to address the limitation of the model of "one size fits all" in most existing searchable encryption schemes, we propose the PRSE_1 scheme to realize personalized rank search. Moreover, based on PRSE_1, we propose the PRSE_2 scheme to tackle the problem of exact keyword search so as to realize the goal of query semantic extension. The encryption strategy we adopt here is based on secure

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

inner product [14], which was initially proposed in [22] for efficient secure nearest neighbor search in cloud. By using this encryption strategy, the index I and the trapdoor T are both well protected.

Algorithm 1. Update(U, w, θ)

Require:
U:the original user interest model;
w:the new query word;
θ: an impact factor, fixed as 1;
Ensure:
U':the updated user interest model
if w in U **then**
  update the score of w, $score = 1 + score \times \theta$;
else
  create a new node with score 1 labeling w;
end if
synonym set=GetSynonymSet();
for every synonym w' in synonym set **do**
  if w' in U **then**
    update the score of w', $score = \alpha + score \times \theta$;
  else
    create a new node with score α labeling w';
    add a edge $w - w'$ and label synonym relation;
  end if
end for
hypernym_hyponym set= GetHypernym_hyponymSet();
for every hypernym/hyponym w' in hypernym_hyponym set **do**
  if w' in U **then**
    update the score of w', $score = \beta + score \times \theta$;
  else
    create a new node with score β labeling w';
    add a edge $w - w'$ and label hypernym/hyponym relation;
  end if
end for
meronym_holonym set= GetMeronym_HolonymSet();
for every meronym/holonym w' in meronym_holonym set **do**
  if w' in U **then**
    update the score of w', $score = \gamma + score \times \theta$;
  else
    create a new node with score γ labeling w';
    add a edge $w - w'$ and label meronym/holonym relation;
  end if
end for
return U';

4.1 PRSE_1

4.1.1 PRSE_1 Scheme

Setup. In this initialization phase, the data owner produces the symmetric key SK, including: 1) a $(n+u+1)$-bit randomly generated vector as S, where n is the total number of keywords and u is a random number $(u > 1)$; 2) two $(n+u+1) \times (n+u+1)$ invertible random matrices $\{M_1, M_2\}$. Hence, the symmetric key SK is a 3-tuple $\{S, M_1, M_2\}$.

BuildIndex(D, SK). In this phase, firstly, the data owner builds an index vector p_i for each document D_i, where each dimension is a normalized keyword weight. And then dimension extending procedure will be conducted: every index vector p_i is extended from n dimensions to $(n+u+1)$ dimensions as $\overrightarrow{p_i}$, where the $(n+j)$th $(j \in [1, u])$ entry in $\overrightarrow{p_i}$ is set as a random number $\varepsilon^{(j)}$ and the last entry $(n+u+1)$th is always set as constant 1. Next, splitting procedure can be applied to $\overrightarrow{p_i}$, which splits $\overrightarrow{p_i}$ into two random vectors $\{\overrightarrow{p_i}', \overrightarrow{p_i}''\}$. Note that the vector S functions as a splitting indicator. Namely, if $S[j]$ is 0, $\overrightarrow{p_i}'[j]$ and $\overrightarrow{p_i}''[j]$ are set as the same value as $\overrightarrow{p_i}[j]$; if $S[j]$ is 1, $\overrightarrow{p_i}'[j]$ and $\overrightarrow{p_i}''[j]$ can be set as random values while the sum of them should be equal to $\overrightarrow{p_i}[j]$. At this moment, the encrypted subindex is built as $I_i = \{M_1^T \overrightarrow{p_i}', M_2^T \overrightarrow{p_i}''\}$. Finally, the data owner outsources the searchable index I and encrypted document collection C onto the cloud.

BuildUIM. In order to realize the personalized search, we should collect and analyze the user's search history to build the user interest model U. Here the user interest model records historical access frequency of query keywords as keyword priority. When a new query keyword w comes, we should update the user interest model as Algorithm 1.

GenQuery$(\widetilde{W}, U, SK)$. This algorithm contains three steps as follows:

[step 1] Search in the user interest model

This step is to get the priority of query words in the user interest model. For each keyword in $\widetilde{W}$, we firstly search for it in the user interest model U. If it exists in U, we just acquire the corresponding node's score; if it doesn't exist in U, we just give it score 1. And then the query q is generated where each dimension is the corresponding score (keyword priority).

[step 2] Priority reformulation

Definition 1 (Priority order of documents to a trapdoor). *For two documents D_i and D_j, D_i is prior to D_j to the trapdoor T if the following condition exists: There exists a keyword priority q_m, such that $p_{i,m} > p_{j,m}$. For other priorities in the query, if there exists a keyword priority q_z such that $q_z > q_m$, then there exist each $p_{i,z}$ in p_i and $p_{j,z}$ in p_j, such that $p_{i,z} = p_{j,z}$.*

Here we note q_m as "*critical priority value* (CPV)" between D_i and D_j.

The accumulated product of the keyword weight and the keyword priority as the relevance score may not return results strictly following the user's search priority.

For example, there are two documents D_1 and D_2 containing only one keyword, w_1 and w_2, respectively where the corresponding keyword weights are $p_{1,1} = 0.3$ and $p_{2,2} = 0.5$. And there is a search request containing two keywords w_1 and w_2 with the keyword priorities $q_1 = 3$ and $q_2 = 2$. The score of D_1 $(0.3 \times 3 = 0.9)$ is less than the score of D_2 $(0.5 \times 2 = 1)$, thereby resulting unexpected rank order. Therefore, we must conduct priority reformulation before deriving the trapdoor so that the returned ranked results strictly adhere to user's search priority.

To reformulate the priorities, besides the symmetric key SK, the data owner should keep additional information, i.e., $\tau = Min\{p_{i,z} - p_{j,z}\}_{1 \le i,j \le m, i \ne j, 1 \le z \le n}$ $(\tau > 0)$. The data owner reformulates the keyword priorities by taking the following steps. Firstly, the data owner sorts the original

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

keyword priorities among the query q in a ascending order where $q_i \leq q_j$ $(i < j)$. Then, we set $q'_1 = q_1$. For q_j, if $q_j > q_{j-1}$, then set $q'_j = \frac{1+\sum_{z=1}^{j-1} q_z}{\tau}$; if $q_j = q_{j-1}$, then $q'_j = q'_{j-1}$. Finally, we reorder the reformulated keyword priorities back to the original position and get the updated query q'.

An example. Suppose that there exists a search query $q = (3, 4, 1, 2)$ after getting the keyword priorities in U, where $\tau = 0.2$. Firstly, we sort the query q into $(1, 2, 3, 4)$. Then we can conduct the priority reformulation, $q'_1 = q_1 = 1$, $q'_2 = (1 + q'_1)/\tau = 10$, $q'_3 = (1 + q'_1 + q'_2)/\tau = 60$, $q'_4 = (1 + q'_1 + q'_2 + q'_3)/\tau = 360$. Finally, we reorder them back into the original format and get the updated query $q' = (60, 360, 1, 10)$.

After the priority reformulation, the document containing the higher priority will always precede others without being affected by the keyword weight, which is proved by Theorem 1.

Theorem 1. *If D_i is prior to D_j to the trapdoor T, then the relevance score of D_i is larger than that of D_j.*

Proof. For two documents D_i and D_j, assume the CPV between D_i and D_j is q_m. Suppose there exists a keyword priority q_z, then the difference of the relevance score between D_i and D_j will be $\sum_z p_{i,z} \cdot q_z - \sum_z p_{j,z} \cdot q_z = \sum_{q_z > q_m} (p_{i,z} - p_{j,z}) \cdot q_z + (p_{i,m} - p_{j,m}) \cdot q_m + \sum_{q_z < q_m} (p_{i,z} - p_{j,z}) \cdot q_z \geq (p_{i,m} - p_{j,m}) \frac{1+\sum_{p_z < p_m} q_z}{\tau} - \sum_{p_z < p_m} p_{j,z} \cdot q_z > 0$ □

[step 3]. Query encryption

After priority reformulation, we get the updated search query q' as q where each dimension represents the corresponding keyword priority. Subsequently, query q is firstly extended to $(n + u + 1)$ dimensions as $\overrightarrow{q}$. For the locations from $(n + 1)$ to $(n + u)$, by randomly choosing v out of u dummy keywords, the corresponding entries are set as 1 and the remaining entries are set as 0. The value of the last entry $(n + u + 1)$th is set as a random number t $(t \neq 0)$ and all the other locations are multiplied by another random number r $(r \neq 0)$. Next, $\overrightarrow{q}$ is split into two random vectors $\{\overrightarrow{q}', \overrightarrow{q}''\}$ following the similar splitting procedure above. The difference is that if $S[j]$ is 0, $\overrightarrow{q}'[j]$ and $\overrightarrow{q}''[j]$ are set as two random numbers so that their sum is equal to $\overrightarrow{q}[j]$; if $S[j]$ is 1, $\overrightarrow{q}'[j]$ and $\overrightarrow{q}''[j]$ are set the same as $\overrightarrow{q}[j]$. After that, the encrypted query vector is generated as the trapdoor $T = \{M_1^{-1}\overrightarrow{q}', M_2^{-1}\overrightarrow{q}''\}$. Finally, the user sends the trapdoor with an optional parameter k to the cloud.

SearchIndex(I, T, k). With the trapdoor T, the cloud server uses "secure inner product" to compute the relevance scores of each document as the following equation. After sorting all relevance scores, the cloud server picks up first k results with the highest scores as $R_{T,k}$ to the user.

$$I_i \cdot T = \{M_1^T \overrightarrow{p_i}', M_2^T \overrightarrow{p_i}''\} \cdot \{M_1^{-1}\overrightarrow{q}', M_2^{-1}\overrightarrow{q}''\} = \overrightarrow{p_i}' \cdot \overrightarrow{q}' + \overrightarrow{p_i}'' \cdot \overrightarrow{q}'' = \overrightarrow{p_i} \cdot \overrightarrow{q} = r(p_i \cdot q + \sum \varepsilon_i^{(v)}) + t.$$

4.1.2 Functionality Analysis

PRSE_1 scheme has well realized the goal of personalized ranked keyword search over encrypted cloud data according to search history. That means different users can achieve personalized ranked results even for the same query according to their respective interests.

4.2 PRSE_2

Due to the fact that most of existing searchable encryption schemes support only exact keyword search, it is possible that the user just gets a few results by querying some terms . In addition, most users are casual when querying some terms in a "hit-or-miss" fashion, or they may hold a viewpoint-"give a try of just randomly picking up some query terms not matching their true search intentions". In such cases, the returned results are definitely not their ideal results. To address this problem, we propose the PRSE_2 scheme with a deep understanding of user's search intention according to their query terms based on PRSE_1, which will return more results as similar as possible to query terms.

4.2.1 PRSE_2 Scheme

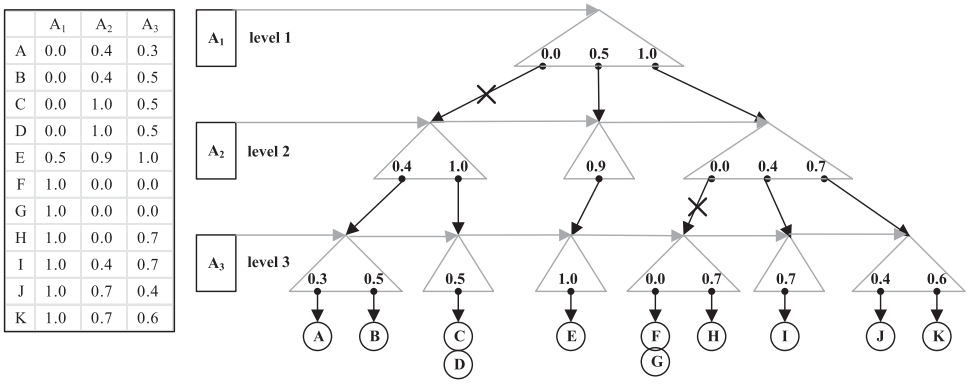
Since our user interest model is built on user's long-term search history, it is made of keywords and their relations. If two words are related with each other, they must have a kind of relation (synonym, hypernym/hyponym, meronym/holonym) in the user interest model. Through our user interest model, we can get some related words to original query terms and add them into the query. In this scheme, the algorithms of *Setup*, *BuildIndex* and *SearchIndex* are the same as those in PRSE_1. The improved details in PRSE_2 scheme are mainly manifested in step 1 (search in the user interest model) of algorithm *GenQuery* as follows. Here we should conduct query semantic extension during step 1.

For each keyword w_i in $\widetilde{W}$ containing t terms, we search for it in the user interest model U. Besides attaining the score of node of corresponding query term, we also construct related keyword set $S_{w_i,d}$ with distance d. Since our user interest model is a diagram of relations in a sense where any relation label means the distance between nodes located at both ends is 1. The distance $dis(w_1, w_2)$ between two words w_1 and w_2 is the number of labels in the shortest path from one of them to the other. The intuitive way to construct the related keyword set of w_i is to put the corresponding node in U as a center and then extend out step by step, enumerating all the possible words w'_i that satisfy the distance criteria $dis(w_i, w'_i) \leq d$. And then we divide the related keyword set $S_{w_i,d}$ into two parts $\{S_{w_i,1}, S_{w_i,2-d}\}$: $S_{w_i,1}$ with distance 1 and $S_{w_i,2-d}$ with distance ranging from 2 to d. For all words in $\{S_{w_i,1}\}_{1 \leq i \leq t}$, we directly get their corresponding priorities and add them to the final query.

Question. With the increasing of the distance parameter d, the related keyword set $S_{w_i,2-d}$ will grow and may has little relationship with w_i or the whole query $\widetilde{W}$. So how to pick out the words in $\{S_{w_i,2-d}\}_{1 \leq i \leq t}$ semantically associated with the whole query $\widetilde{W}$?

Solution. Note that the search query $\widetilde{W}$ usually represents the tendency of query. We can consider it as a point in a semantic space. The words in $\{S_{w_i,2-d}\}_{1 \leq i \leq t}$ as the candidate set should be closer to this point. That is, the words to be

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.



	A_1	A_2	A_3
A	0.0	0.4	0.3
B	0.0	0.4	0.5
C	0.0	1.0	0.5
D	0.0	1.0	0.5
E	0.5	0.9	1.0
F	1.0	0.0	0.0
G	1.0	0.0	0.0
H	1.0	0.0	0.7
I	1.0	0.4	0.7
J	1.0	0.7	0.4
K	1.0	0.7	0.6

Fig. 3. Set of seven objects with values of three attributes stored in MDB-tree.

added to the final query should be more similar to the whole original query $\widetilde{W}$. We set the following criteria to choose the word w meeting the requirements:

$$overlay(\{S_{w_i,2-d}\}_{1\le i\le t}, w) \ge \left\lfloor \frac{t}{2} \right\rfloor + 1, \tag{2}$$

where $overlay(\{S_{w_i,2-d}\}_{1\le i\le t}, w)$ means how many related keyword sets the candidate term w exist in. $\lfloor \frac{t}{2} \rfloor$ will always round down to the nearest whole unit to $\frac{t}{2}$. Specifically, the candidate term w should exist in at least half of the related keyword sets. For all the terms meeting the criteria above, we can also get their corresponding priorities and add them to the final query. After that, the final query is produced. Next steps are the same as those in PRSE_1.

4.2.2 Functionality Analysis

In PRSE_2, we effectively realize query semantic extension according to original query terms. Thus, when inputting same query terms, the user will get more related results than PRSE_1 with big k. Therefore, this scheme effectively solves the problem of exact keyword search.

5 The Efficiency Improvement of PRSE

In the last section, we present a solution that requires the trapdoor T to compute the relevance score with each document stored in cloud. To improve its search efficiency and thus to scale well for large data sources, in this section we find a technique to optimize the solution. Rather than some clustering algorithms and local sensitive hash techniques putting some similar documents together, we adopt a MDB-tree [31] to construct the searchable index to address the top-k problem, which has been applied in [32] earlier.

MDB-tree. MDB-tree allows to index a set of objects by attributes $A_1, \ldots, A_m$, $m > 1$ in one data structure, as shown in Fig. 3. MDB-tree has m levels and each level represents an attribute domain. Each node is a B^+-tree. Note that each level has a prediction threshold value $\widehat{V}_i$, as an important search parameter, which is obtained from the maximum value V_i at each level. For example, in Fig. 3, $\widehat{V}_1 = \widehat{V}_2 = \widehat{V}_3 = 1.0$. In the top-k search procedure, it adopts a depth-first method to start from the root node. In the search process downward, it always chooses the node with the unused highest value, predicting the maximum possible final score to be obtained. If the predicted final score is less than the minimum score among the top-k objects chosen, search process will return to its parent, otherwise, it will go down to the child node at the next level. The search process is performed recursively until the top-k objects are produced. With the prediction threshold value, search process doesn't need to go through the whole tree and only accesses a part of nodes actually. This "incomplete traversing" saves a lot of operating time. Fig. 3 shows an example: when k=3, only the nodes, E, J and K, are returned to the user. The cross sign in the figure means search process doesn't need to go through the path.

In order to apply this MDB-tree into this scheme, we divide the encrypted subindex I_i in **BuildIndex**(D, SK) into multiple sub-vectors that each sub-vector $I_{i,j}$ corresponds a subset of keywords and represents the jth level of index tree, as shown in Fig. 4. The trapdoor T in **GenQuery**$(\widetilde{W}, U, SK)$ is divided in the same way I_i done. Through the adjustment above, the searchable index can be transformed into a MDB-tree in Fig. 4. Therefore, top-k objects can be produced by the tree-based search algorithm. As mentioned above, the important factor to affect the search efficiency is the prediction threshold value $\widehat{V}_j$ at each level. To produce a better prediction threshold $\widehat{V}_j$, the data owner holds an auxiliary vector E_j during index construction, which consists of the maximum value at each dimension at this level. Subsequently, during query generation, $\widehat{V}_j$ is equal to the inner product of E_j and T_j. The prediction threshold $\widehat{V}_j$ will be sent together with the trapdoor T to the cloud server. During the search process, the main idea is that if the predicted final score $(\sum_{j=1}^{j=J} I_{i,j} \times T_j + \sum_{j=J+1}^{j=h} \widehat{V}_j)$ of the node at the Jth level is less than the minimum score of the top-k objects chosen, it will stop going down to the next level.

6 Security Definitions and Privacy Analysis

In this section, we will discuss and analyze the security of PRSE. The security of this scheme depends upon secure *kNN* [22] which mainly contains three steps: matrix encryption, random split and dimension extension.

So before stating our definitions for PRSE, we introduce some basic idea of KNN firstly. Given a query, the main

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

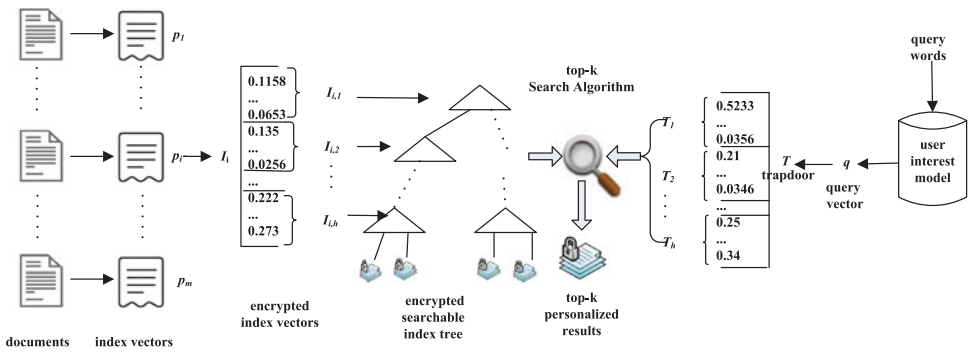


Fig. 4. Improvement of PRSE scheme.

focus of *kNN* problem is to compute the "distance differential". That is, is p_2 closer to q than p_1?

$$\begin{aligned} kNN \Leftrightarrow & \|p_1 - q\| - \|p_2 - q\| > 0 \\ & \Leftrightarrow (p_1 \cdot p_1 - 2p_1 \cdot q) - (p_2 \cdot p_2 - 2p_2 \cdot q) > 0 \\ & \Leftrightarrow ((p_2, -0.5(p_2 \cdot p_2)) \cdot (q, 1) > ((p_1, -0.5(p_1 \cdot p_1) \cdot (q, 1) \\ & \Leftrightarrow p_2^* \cdot q^* > p_1^* \cdot q^* \\ & \Leftrightarrow E(p_2^*) \cdot E(q^*) > E(p_1^*) \cdot E(q^*), \end{aligned}$$

where $p^* = (p, -0.5(p \cdot p)), q^* = (q, 1)$, $\|p\|$ represents euclidean norm of p, $\cdot$ is inner product, and $E(\cdot)$ is an encryption function.

Note that, according to the calculation of keyword weight in our paper, each $\|p\| = 1$. Hence, we can also directly get the formula as follows:

$$\begin{aligned} kNN \Leftrightarrow & \|p_1 - q\| - \|p_2 - q\| > 0 \\ & \Leftrightarrow p_2 \cdot q > p_1 \cdot q \\ & \Leftrightarrow E(p_2) \cdot E(q) > E(p_1) \cdot E(q). \end{aligned}$$

One common approach to securely support *kNN* is distance-preserving transformation (DPT) that encrypts index points so that the distance between any two encrypted points is same as that between the corresponding points in the plain index database. DPT can resist level-1 attack. Unfortunately, it is shown to be not secure in practice.

6.1 Security Definitions

Definition 2 (Distance-recoverable Encryption (DRE)). *Given an encryption function E and a key K, let $E(p, K)$ be the encrypted value of a point p in a plain index database. E is distance recoverable if and only if there exists a computational procedure f such that $\forall p_1, p_2, K, f(E(p_1, K), E(p_2, K)) = d(p_1, p_2)$.*

We observe that the weakness of DRE is the fact that the cloud server can recover distance information from the encrypted index database. More specifically, the distance $d(p_1, p_2)$ is determined by their encrypted values $E(p_1, K)$ and $E(p_2, K)$. The leakage of distance information allows the cloud server to compute signatures and thus to apply the signature linking attack [22]. Therefore, no DRE (e.g., DPT) can survive level-3 attack and it has poor resistance to level-2 attack.

Definition 3 (Matrix encryption). *To resist level-2 attack, we need an encryption function that reveals no distance information between any two index points p_i and p_j. That is, it is not distance-recoverable. To address the problem of distance-recoverable, the scheme based on matrix encryption is promoted. We summarize the procedures of matrix encryption scheme as follows.*

- Key*: a $n \times n$ invertible matrix M.*
- Index encryption*: The encrypted index point $\widehat{p} = M^T p$.*
- Query encryption*: The encrypted query point $\widehat{q} = M^{-1} rq$, where r is a random number.*
- Distance comparison*: To determine whether p_2 is closer to q than p_1, it only needs to check whether $(\widehat{p_2} - \widehat{p_1}) \cdot \widehat{q} > 0$, where $\widehat{p_1}$, $\widehat{p_2}$ and $\widehat{q}$ are encrypted values of p_1, p_2, q respectively.*

Theorem 2. *Suppose $\widehat{p_1}$, $\widehat{p_2}$ and $\widehat{q}$ are the encrypted values of the index points p_1, p_2 and the query point q, respectively, matrix encryption scheme correctly determines whether p_2 is closer to q than p_1 by checking $(\widehat{p_2} - \widehat{p_1}) \cdot \widehat{q} > 0$.*

Proof. Note that,

$$\begin{aligned} (\widehat{p_2} - \widehat{p_1}) \cdot \widehat{q} &= (\widehat{p_2} - \widehat{p_1})^T \widehat{q} \\ &= (p_2 - p_1)^T rq. \end{aligned}$$

So, the condition is equal to

$$\begin{aligned} (p_2 - p_1)^T rq > 0 \Leftrightarrow & p_2^T q > p_1^T q \\ & \Leftrightarrow \|p_1 - q\| > \|p_2 - q\| \\ & \Leftrightarrow d(p_1, q) > d(p_2, q). \end{aligned}$$

□

Theorem 3. *Matrix encryption is not distance-recoverable.*

Proof. According to our definition, if an encryption E is distance-recoverable, given any encryption values a_1 and a_2, the distance $d(p_1, p_2)$ can be computed by $f(a_1, a_2)$ regardless the encryption key K where $a_1 = E(p_1, K_1)$ and $a_2 = E(p_2, K_1)$. In this matrix encryption scheme, it is possible to construct two index points x and y and an encryption key K_2 such that

1. $E(x, K_2) = a_1 = E(p_1, K_1)$, and
2. $E(y, K_2) = a_2 = E(p_2, K_1)$, but
3. $d(x, y) \neq d(p_1, p_2)$.

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

Now, if E is a DRE, we have
$f(a_1, a_2) = f(E(p_1, K_1), E(p_2, K_1)) = d(p_1, p_2)$ and
$f(a_1, a_2) = f(E(x, K_2), E(y, K_2)) = d(x, y)$.
It leads a contradiction since $d(x, y) \neq d(p_1, p_2)$. □

This matrix encryption scheme does not reveal any distances between index points and thus it prevents level-2 attack. However, this scheme is not secure against level-3 attack.

Theorem 4. *Assume the matrix encryption scheme is attacked by a level-3 cloud server who has knowledge $H =< C, I, T, P_A, I_A >$. If there are n index points p_i $(1 \leq i \leq n)$ in P_A such that the vectors p_i are linearly independent, then the cloud server can recover the whole plain index P.*

Proof. To recover the plain index P, the cloud server needs to recover the encryption key - a $n \times n$ matrix M. Since we have $P_A = \{p_1, \ldots, p_n\}$ and the corresponding encrypted values I_i, we can set up the following equations to solve M: $M^T \cdot p_i = I_i$ from $i = 1$ to n. since the p_i are linearly independent, we can easily compute M. Hence, we are able to recover the whole plain index P from the encrypted index I. □

Definition 4 (Random split). *To address the limitation of matrix encryption scheme above, it just needs to make the matrix M harder to crack. One popular method is to introduce some randomness, making it difficult to set up the equations. Combined with random split, the advanced scheme based on matrix encryption is summarized as follows.*

- Key*: two $n \times n$ invertible matrix M_1, M_2; a n-bit randomly generated vector as S.*
- Index encryption*: The index point p is firstly split into two parts $\{p', p''\}$. If $S[j]$ is 0, $p'[j]$ and $p''[j]$ are both set as $p[j]$; otherwise, $p'[j]$ and $p''[j]$ can be set as random values while the sum of them should be equal to $p[j]$. The encrypted value of p is the pair $\widehat{p} = \{M_1^T p', M_2^T p''\}$.*
- Query encryption*: The query point q is also firstly split into two parts $\{q', q''\}$. If $S[j]$ is 0, $q'[j]$ and $q''[j]$ can be set as random values so that the sum of them should be equal to $q[j]$; otherwise, $q'[j]$ and $q''[j]$ are both set as $q[j]$; The encrypted value of q is the pair $\widehat{q} = \{M_1^{-1} q', M_2^{-1} q''\}$.*
- Distance comparison*: To determine whether p_2 is closer to q than p_1, it only needs to check whether $(\widehat{p_2} - \widehat{p_1}) \cdot \widehat{q} > 0 \Leftrightarrow (p_2 - p_1) \cdot q > 0$, where $\widehat{p_1}$, $\widehat{p_2}$ and $\widehat{q}$ are encrypted values of p_1, p_2, q respectively.*

Theorem 5. *Owing to the introduction of random split, the scheme can resist the level-3 attack if the cloud server cannot recover the splitting configuration (e.g., the vector S in key).*

Proof. According to the definition, for any index point $p_i \in P_A$, a level-3 cloud server knows the encrypted values $I_i = \{\widehat{p_{ia}}, \widehat{p_{ib}}\} = \{M_1^T p_i', M_2^T p_i''\}$. If the cloud server doesn't know splitting configuration, he has to set p_i' and p_i'' as two random n-dimension vectors. And then he sets the following equations: $M_1^T p_i' = \widehat{p_{ia}}$ and $M_2^T p_i'' = \widehat{p_{ib}}$, where M_1 and M_2 are two $n \times n$ matrix. There are $2n|P_A|$ unknown variables in p_i' and p_i'' and $2n^2$ unknown variables in M_1 and M_2. Since there are only $2n|P_A|$ equations, the cloud server doesn't have sufficient information to crack the matrix. Hence, random split based on matrix encryption can resist the level-3 attack. □

Basically, in order to solve the matrix, the cloud server has to try out all splitting configuration. Since there are 2^n possible splitting configurations, the introduction of random split makes the scheme 2^n more costly to attack compared with the scheme only based on matrix encryption. Given $n = 80$, if a cloud server can search $1T$ configurations per second, it requires $38k$ years to try all possible configurations. Hence, if n is large enough, it is impossible to recover the plain data using this advance scheme.

Definition 5 (Dimension extension). *Until now, given a certain query q, for each index point p_i, the distance between p_i and q is fixed and accurate relatively. According to [14], this deterministic property makes keyword privacy be in danger. The cloud server can easily deduce whether a certain keyword exists in a document by analyzing the correlation of some trapdoors. In consideration of this, through extending dimension to increase some dummy entries, the keyword privacy and trapdoor unlinkability will be protected.*

6.2 Privacy Analysis

Next, we analyze PRSE concerning the search privacy requirements as described in Section 2.

Keyword privacy. Because of the introduction of random number $\varepsilon^{(j)}$ in each index and random locations of the dummy keywords upon each search request, $\sum \varepsilon_i^{(v)}$ as the part of the final relevance score will be different even to the same document. In consideration of search accuracy, every $\varepsilon^{(j)}$ follows the same uniform distribution $U(\mu' - c, \mu' + c)$ where the mean is μ' and the variance is $c^2/3$ as σ'^2. According to the central limit theorem, the sum of v independent random values $\varepsilon^{(j)}$ follows the normal distribution $N(\mu, \sigma^2)$, where the mean μ is $v\mu'$ and the variance σ^2 is $vc^2/3$. Therefore, we can generate the $\varepsilon^{(j)}$ based on the value of $\mu' = \mu/v$ and the value of $c = \sqrt{3/v}\sigma$. The standard deviation σ functions as a flexible trade-off parameter between security and search accuracy. Note that σ should be small from the effectiveness point of view, but it will enable the cloud to get more statistic information of the original scores, which also weakens the trapdoor unlinkability. Therefore, σ should be large enough for security reason.

index confidentiality and query confidentiality. As for index confidentiality, with the introduction of dummy keywords and randomly splitting, each keyword weight within index is encrypted by the secret matrices. Therefore, the cloud server will find it difficult to deduce the meaning of every dimension in the index, which has been proved above. Based on the same principle, the keyword priorities in the query will be invisible to the cloud server, thus query confidentiality can be protected.

Trapdoor unlinkability. With the random choice of locations of dummy keywords and the introduction of some random numbers (e.g., r and t), our PRSE scheme can generate two different trapdoors even for the same search request. With different trapdoors, different relevance scores will be

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

produced even for the same document, which will make the cloud server have trouble mining the relationship between two trapdoors by comparing them directly. This nondeterministic encryption guarantees the trapdoor unlinkability, which is an urgent problem to be solved in many symmetric key based searchable encryption schemes [15].

7 Performance Evaluation

In this section, we present thorough performance analysis of our proposed search scheme over encrypted data. We implement all the algorithms mentioned in our paper on a 2.83 GHZ Intel Core(TM) processor, Windows 7 operating system with a RAM of 4 GB. In our experiments, we choose a publicly available real dataset: the Enron Dataset [27] and also try to get the root of every keyword with a well known stemming technique called Porter Stemming Algorithm [30]. Rest of this section is organized as follows. We describe how to set up the experiment firstly, and then valuate the efficiency compared with MRSE [14]. At last, we discuss the complexity analysis and compare it with MRSE in theory.

As we mentioned above, we use the Enron Dataset as our corpus for experiment. However, there are a large number of duplicated emails in the original dataset, so we use the MySql database [28] provided by Jitesh et al., which is created upon the Enron Dataset without duplicates. The details of database can be found in [29]. We choose 10,000 records randomly as our final experiment corpus. In order to extract the keywords which can represent the email well and improve our search accuracy, we should process every word in the email through a series of steps, including getting rid of the stop-words, converting each word into lowercase and returning each word to its root (stemming). In the keyword extraction, for each email, we compute the TF×IDF of each word according to formula 1, sort them in descending order and only pick up first five number of words as the keywords of the corresponding email. At last, we collect the distinct keywords in all the emails as our keyword dictionary (18,711 keywords).

7.1 Precision

In this section, we mainly focus on the precision of our scheme. The precision here means whether users attain the desired documents according to queries or not. In our scheme, we also need balance precision and privacy. To reduce the loss of precision, balance strategy of precision and privacy by introducing of dummy keywords in [14] is used in our scheme. In theory, It's highly possible to return all the query results related to the users' recent search results because of the construction of users' interesting model (keywords in users' search history have high-priority). That brings great convenience to users who major on a certain subject recently and has a great impact on users' work. To verify the precision of our scheme, we conduct a survey that 20 users are randomly invited to make a test for our system. As might be expected, more than three fourths are satisfied with the returned results by our system. This also proves the effectiveness of our system. We also conduct a survey on our documents set to perform top-k (we set k=20) search, and all the 20 returned documents accord with our query. The first one of the returned results is what we need. However, in the documents set, there are 100 documents related to our query. The recall rate is 0.2, but the most related documents are returned. Therefore, it is a lack of practical significance to discuss the recall rate.

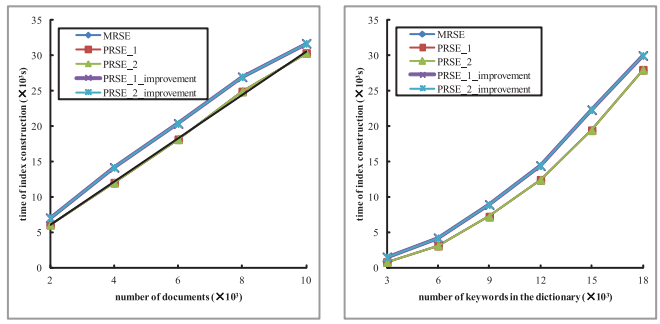


Fig. 5. Time of index construction: (a) For the different size of document collection with the same keyword dictionary, $n = 18,711$; (b) For the different size of keyword dictionary with the same document collection, $m = 10,000$.

7.2 Efficiency

7.2.1 Index Construction

In the basic PRSE scheme, we need to build a searchable subindex I_i for each document D_i in the document collection. The first step is to map the keywords extracted from D_i into the keyword set to generate a data vector p_i, and then encrypt the vector (including extending and splitting procedure). In the PRSE improvement scheme, the data owner also needs to divide the subindex and construct the index tree (MDB-tree). In our experiment, we adopt a method of building a trie based on the keyword set and therefore the time cost of mapping procedure is determined by the length of word. However, the time cost of encryption depends directly on the dimensionality of data vector which is determined by the number n of keyword set W. In addition, the time cost of building the searchable index for the whole document collection depends on the number m of documents in collection D. The process of index construction among MRSE, PRSE_1 and PRSE_2 is nearly same. Due to the construction of index tree in PRSE improvement scheme, the time cost both in PRSE_1 improvement and PRSE_2 improvement scheme is a little more than basic scheme. Fig. 5a shows that, given the same keyword dictionary where $n = 18,711$, the time cost of building the whole index is nearly linear with the number of documents, since the time cost of building a subindex as a major computation is fixed (near to 3s/document). Fig. 5b shows that, given the same document collection where $m = 10,000$, the time cost of building the whole index is determined by the number of keyword dictionary. As presented in Section 3, the major computation in the phase of building index includes the splitting procedure and two multiplications of a $(n+u+1) \times (n+u+1)$ matrix and a $(n+u+1)$ vector, so the complexity is $O((n+u)^2)$, which is verified by the shape of second-degree parabola in Fig. 5b. There is no denying that the operation of index construction is a burden for the data owner. However, it is just a one-time cost before data outsourcing. Besides, we record the storage overhead of

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Storage of Subindex/Query

Size of keyword dictionary	3,000	6,000	9,000	12,000	15,000	18,000
MRSE(KB)	23.44	46.88	70.31	93.75	117.19	140.63
PRSE(KB)	46.88	93.75	140.63	187.5	234.37	281.25

TABLE 2
Storage of User Interest Model

Size of query terms	500	1,000	1,500	2,000	2,500	3,000
UIM(KB)	12.20	23.43	38.08	50.78	63.47	76.17

each subindex in Table 1 with different size of keyword dictionary in comparison with MRSE. The storage of each subindex is linearly related to the dimensionality of index vector, and while the dimensionality is dependent on the number of keywords in the dictionary. Moreover, the storage of all PRSE schemes is nearly twice that of MRSE due to the difference of index structure. The subindex in MRSE is a binary vector, while each dimension in the subindex of PRSE represents a keyword weight containing decimals, just like data structure "int" versus "double".

7.2.2 User Interest Model Construction

In order to evaluate the construction of user interest model, we randomly select three users, analyze their historic records and extract their query terms. Fig. 6 shows the time cost among three users with respect to different size of historic records. We can see the time cost rises with number of historic records with respect to a user. Due to diversity of historic records among the users, their time cost in the same size of historic records varies widely. Besides, we record the storage overhead of user interest model in Table 2 with different size of query terms.

7.2.3 Query Generation

Fig. 7a shows that the time cost to generate a query mainly depends on the number of keywords in the dictionary, since the common main operation or time-consuming operation in all the schemes is query encryption. So the time cost will become large as increasing the number of keywords in the dictionary. Fig. 7b shows the performance of query generation has little relationship with the number of query words in each scheme. Besides the common step of query encryption, our PRSE contains two additional operations compared with MRSE. In PRSE_1, we should search for the query words in the well-built user interest model and then get the score of each word, followed by the priority reformulation. At last, query encryption will be conducted. The difference between PRSE_1 and PRSE_2 is that query semantic extension needs to be carried out during the step of search in the user interest model. In PRSE_2, besides the original query words, we should also seek out some candidate words related to original ones and add them to the final query. Compared with the basic PRSE schemes (PRSE_1 and PRSE2), the improvement schemes (PRSE_1 improvement and PRSE_2 improvement) includes a addition operation: divide the trapdoor T into multiple sub-trapdoors T_i. The time cost of additional operations in all PRSE schemes (search in the user interest model and priority reformulation,trapdoor division) is negligible compared with the operation of query encryption. So the time cost of query generation in all PRSE schemes is a little slower than that of MRSE from Fig. 7b.

7.2.4 Search

The process of search among the schemes of MRSE, PRSE_1 and PRSE_2 is same. It consists of computing and ranking relevance scores for all the data files in the document collection. And the search process in PRSE improvement schemes is comprised of computing and ranking the relevance scores of relevant documents rather than all documents in the tree search. Fig. 8a shows the search time with respect to the different size of document collection. From the figure, we can see our improvement PRSE schemes are far efficient than MRSE and basic PRSE schemes. Fig. 8b shows the our improvement schemes are quite efficient when users request more relevant documents.

8 Related Work

8.1 Searchable Encryption

The first practical searchable encryption scheme in symmetric setting is proposed by Song et al. [1], in which each word is encrypted independently under a two-layer construction

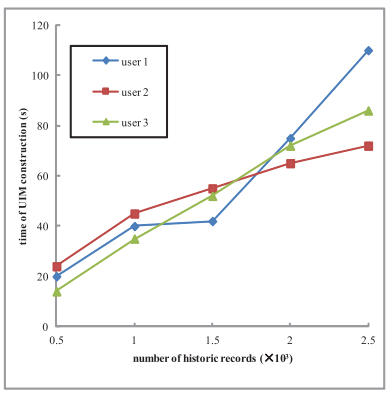


Fig. 6. Time of UIM construction.

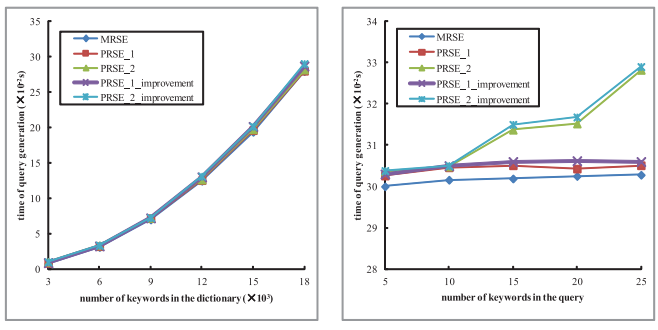


Fig. 7. Time of query generation: (a) For the different size of keyword dictionary with the same query words, $t = 10$; (b) For the different number of query words with the same keyword dictionary, $n = 18,711$.

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

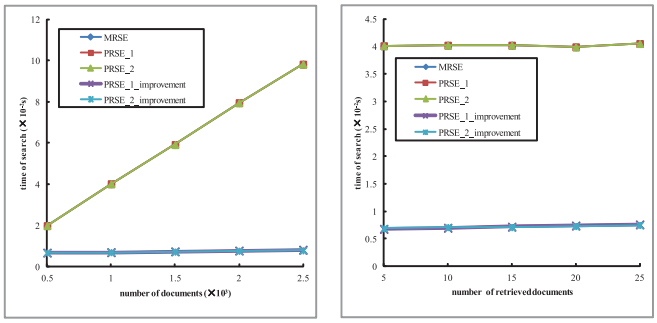


Fig. 8. Time of search: (a) For the different size of document collection with the same dictionary, $n = 18,711$, $k = 10$; (b) For the different number of retrieved documents with the same document collection and dictionary, $m = 1{,}000$, $n = 18{,}711$.

and the user has to go through the whole document to search a certain keyword. And then some security definitions and many improvements or constructions have been proposed by Goh [2], Chang and Mitzenmacher [3] and Curtmola et al. [4]. Boneh et al. [5] propose the first public key-based searchable encryption scheme, where anyone owning the private key can search data items encrypted by the public key. Recently, Cash et al. [34] and Stefanov et al. [35] discuss the problem of information leakage in conjunctive search and dynamic searchable symmetric encryption, respectively. After that, a lot of schemes [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] under different applications have been proposed. Among them, to address the spelling mistake, import issue of fuzzy keyword search is proposed by Li et al. [11] and improved in [12]. And Wang et al. [13] and Cao et al. [14] do research on secure ranking on single keyword and multi-keyword respectively. After that, Sun et al. [32] improve the efficiency of multi-keyword search by adopting MDB-tree. However, most of existing searchable encryption schemes support only exact keyword search, which affects data usability and user's experience. Fu et al. [33] propose a semantic keyword search scheme based on stemming algorithm, which helps users find relevant documents containing semantically close keywords related to the query word. Furthermore, personalized search is also missed or ignored. Shen et al. [19] proposed a preferred keyword search scheme over encrypted data, but how to measure keyword preference is ignored. The artificial method of measuring keyword preference is time consuming and imposes a burden on the user. Moreover, it fails to consider different users' search histories and thus has great randomness.

8.2 Personalized Search

Personalized search aims at exploiting user information to enable search results better meet the individual user's search intention. The general approach is to build a user profile, which describes the user's interests or preferences that can directly set by the user or collected during the search history for a period. And the most challenging works in personalized search are: 1) how to build a user profile model; 2) how to make use of the user profile to improve the search. The user profile model is often built upon a set of keyword vectors or classes of keyword vectors. Due to the absence of interrelation of keywords in the keyword-based representation model, some researchers make use of a set of concepts derived from predefined ontology or reference ontology to express the user profile model. It is well known that ontology-based user profile model is superior to other representation models as it can exploit semantic knowledge. Note that search personalization is achieved by integrating the user profile in the query reformulation process [23], query-document matching [24], personalized document categorization [25] or document re-ranking [26]. The current search engines, such as GOOGLE, has launched personalized search, where the user can indicate his interests or preferences explicitly, or preferences may be automatically gathered. However, they have not been widely adopted yet since users worry about the leakage of user privacy. Moreover, most of existing personalized search schemes are inapplicable to ciphertext.

9 Conclusion

In this paper, we address the problem of personalized multi-keyword ranked search over encrypted cloud data. Considering the user search history, we build a user interest model for individual user with the help of semantic ontology WordNet. Through the model, we have realized automatic evaluation of the keyword priority and solved the limitation of the artificial method of measuring. Moreover, we propose two PRSE schemes to solve two limitations (the model of "one size fit all" and keyword exact search) in most existing searchable encryption schemes. In addition, thorough privacy analysis and performance analysis demonstrates that our scheme is practicable.

Acknowledgments

This work is supported by the National Science Foundation of China (61373133, U1536206, 61232016, U1405254, 61502242), BK20150925, PAPD fund, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Prospective Research Project on Future Networks of Jiangsu Future Networks Innovation Institute (BY2013095-4-10), and US National Science Foundation(NSF) under grant CNS-1262277.

References

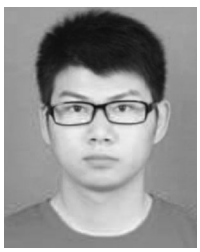
[1] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, 2000, pp. 44–55.
[2] E.-J. Goh. (2003). Secure indexes. Cryptology ePrint Archive, Rep. 2003/216 [Online]. Available: http://eprint.iacr.org/
[3] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. 3rd Int. Conf. Appl. Cryptography Netw. Security*, 2005, pp. 442–455.
[4] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 79–88.
[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2004, pp. 506–522.
[6] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Trans. Commun.*, vol. E98-B, no. 1, pp. 190–200, 2015.

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

[7] Z. Xia , X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, DOI: 10.1109/TPDS.2015.2401003, 2015.
[8] Z. Fu, X. Sun, N. Linge, and L. Zhou, "Achieving effective cloud search services: Multi-keyword ranked search over encrypted cloud data supporting synonym query," *IEEE Trans. Consumer Electron.*, vol. 60, no. 1, pp. 164–172, Feb. 2014.
[9] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. 4th Conf. Theory Cryptography*, 2007, pp. 535–554.
[10] Z. Fu, X. Sun, Z. Xia, L. Zhou, and J. Shu, "Multi-keyword ranked search supporting synonym query over encrypted data in cloud computing," in *Proc. IEEE 32nd Int. Perform. Comput. Commun. Conf.*, San Diego, CA, USA, 2013, pp. 1–8.
[11] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. J. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, 2010, pp. 1–5.
[12] C. Liu, L. H. 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.*, 2011, pp. 269–273.
[13] C. Wang, N. Cao, J. Li, K. Ren, and W. J. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst. Workshop*, 2010, pp. 253–262.
[14] N. Cao, C. Wang, M. Li, K. Ren, and W. J. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, 2011, pp. 829–837.
[15] M. Chuah and W. Hu, "Privacy-aware BedTree based solution for fuzzy multi-keyword search over encrypted data," in *Proc. 31st Int. Conf. Distrib. Comput. Syst. Workshops*, 2011, pp. 273–281.
[16] C. Wang, K. Ren, S. C. Yu, and K. M. R. Urs, "Achieving usable and Privacy-assured similarity search over outsourced cloud data," in *Proc. IEEE INFOCOM*, 2012, pp. 451–459.
[17] M. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. NDSS Symp.*, 2012, https://www.internetsociety.org/sites/default/files/06_1.pdf
[18] C. Liu, L. Zhu, M. Wang, and Y. Tan. (2013). Search pattern leakage in searchable encryption: Attacks and new constructions," Cryptology ePrint Archive, Report 2013/163 [Online]. Available: http://eprint.iacr.org/
[19] Z. Shen, J. Shu, and W. Xue, "Preferred keyword search over encrypted data in cloud computing," in *Proc. IEEE 21st Int. Symp. Quality Service*, 2013, pp. 1–6.
[20] G. A. Miller, "WORDNET: A lexical database for english," *Commun. ACM*, vol. 2, no. 11, pp. 39–41, 1995.
[21] B. Chor, O. Goldreich, E. Kushelivitz and M. Sudan, "Private information retrieval," in *Proc. 36th IEEE Conf. Found. Comput. Sci.*, 1995, pp. 41–50.
[22] 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.
[23] X. Shen, B. Tan and C. Zhai, "Context-sensitive information retrieval using implicit feedback," in *Proc. 28th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2005, pp. 43–50.
[24] W. Fan, M. D. Gordon, and P. Pathak, "Discovery of context-specific ranking functions for effective information retrieval using genetic programming," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 4, pp. 523–527, Apr. 2004.
[25] Z. Ma, G. Pant, and O. R. L. Sheng, "Interest-based personalized search," *ACM Trans. Inf. Syst.*, vol. 25, no. 1, pp. 1–38, 2007.
[26] A. Sieg, B. Mobasher, and R. Burke, "Web search personalization with ontological user profiles," in *Proc. ACM 16th Conf. Inf. Knowl. Manage.*, 2007, pp. 525–534.
[27] (2015, May). Enron Email Dataset [Online]. Available: https://www.cs.cmu.edu/~enron/, last accessed 08-25-2015.
[28] (2015, May). Enron Dataset [Online]. Available: http://www.isi.edu/~adibi/Enron/Enron.htm, last accessed 08-25-2015
[29] (2015, May). The Enron email dataset database schema and brief statistical report [Online]. Available: http://www.isi.edu/~adibi/Enron/Enron_Dataset_Report.pdf, last accessed 08-25-2015.
[30] M. F. Porter, "An algorithm for suffix stripping," *Automated Library Inf. Syst.*, vol. 14, no. 3, pp. 130–137, 1980.
[31] M. Ondreička and J. Pokorný, "Extending fagin's algorithm for more users based on multidimensional B-tree," in *Proc. 12th East Eur. Conf. Adv. Databases Inf. Syst.*, 2008, pp. 199–214.
[32] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting Similarity-based ranking," in *Proc. ACM 8th SIGSAC Symp. Inf., Comput. Commun. Security*, 2013, pp. 71–82.
[33] Z. Fu, J. Shu, X. Sun, and D. Zhang, "Semantic keyword search based on trie over encrypted cloud data," in *Proc. Proc. 2nd Int. Workshop Security Cloud Comput.*, 2014, pp. 59–62.
[34] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. (2013). Highly-scalable searchable symmetric encryption with support for boolean queries. Cryptology ePrint Archive, Rep. 2013/169 [Online]. Available: http://eprint.iacr.org/
[35] E. Stefanov, C. Papamanthou, and E. Shi. (2013). Practical dynamic searchable encryption with small leakage. Cryptology ePrint Archive, Rep. 2013/832, 2013 [Online]. Available: http://eprint.iacr.org/

Zhangjie Fu received the PhD degree in computer science from the College of Computer, Hunan University, China, in 2012. He is currently an associate professor at the College of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include cloud and outsourcing security, digital forensics, network and information security. His research has been supported by NSFC, PAPD, and GYHY. He is a member of the IEEE, and a member of ACM.

Kui Ren received the PhD degree from Worcester Polytechnic Institute. He is an associate professor of computer science and engineering and the director in UbiSeC Lab at State University of New York at Buffalo. His current research interest spans cloud and outsourcing security, wireless and wearable system security, and human-centered computing. His research has been supported by the US National Science Foundation (NSF), US Department of Energy (DOE), AFRL, MSR, and Amazon. He received the US National Science Foundation (NSF) CAREER Award in 2011 and Sigma XiIIT Research Excellence Award in 2012. He also received the Best Paper Awards including IEEE ICNP 2011. He currently serves as an associate editor for *IEEE TMC*, *TIFS*, *IEEE Wireless Communications*, *IEEE IoT-J*, *IEEE TSG*, *Elsevier Pervasive and Mobile Computing*, and *Oxford The Computer Journal*. He is a fellow of the IEEE, a member of ACM, and a distinguished lecturer of the IEEE Vehicular Technology Society.



Jiangang Shu received the BE degree in network technology and engineering from Nanjing University of Information Science and Technology (NUIST), Nanjing, China, in 2012. He is currently working toward the MS degree in computer science and technology at the Department of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include cloud security, steganography, network and information security.

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.

Xingming Sun received the BS degree in mathematics from Hunan Normal University, China, in 1984, the MS degree in computing science from Dalian University of Science and Technology, China, in 1988, and the PhD degree in computing science from Fudan University, China, in 2001. He is currently a professor at the Department of Computer and Software, Nanjing University of Information Science and Technology, China. In 2006, he visited the University College London, United Kingdom; he was a visiting professor in University of Warwick, United Kingdom, between 2008 and 2010. His research interests include network and information security, database security, and natural language processing. He is a senior member of the IEEE.

Fengxiao Huang received the BE degree in network technology and engineering from Nanjing University of Information Science and Technology (NUIST), Nanjing, China, in 2014. He is currently working toward the MS degree in computer science and technology at the Department of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include cloud security and information security.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Authorized licensed use limited to: Qingdao University. Downloaded on December 20,2023 at 02:19:27 UTC from IEEE Xplore. Restrictions apply.