Privacy Preserving Ranked Multi-Keyword Search for Multiple Data Owners in Cloud Computing

Wei Zhang, *Student Member, IEEE,* Yaping Lin, *Member, IEEE,* Sheng Xiao, *Member, IEEE,* Jie Wu, *Fellow, IEEE,* and Siwang Zhou

Abstract—With the advent of cloud computing, it has become increasingly popular for data owners to outsource their data to public cloud servers while allowing data users to retrieve this data. For privacy concerns, secure searches over encrypted cloud data has motivated several research works under the single owner model. However, most cloud servers in practice do not just serve one owner; instead, they support multiple owners to share the benefits brought by cloud computing. In this paper, we propose schemes to deal with Privacy preserving Ranked Multi-keyword Search in a Multi-owner model (PRMSM). To enable cloud servers to perform secure search without knowing the actual data of both keywords and trapdoors, we systematically construct a novel secure search protocol. To rank the search results and preserve the privacy of relevance scores between keywords and files, we propose a novel Additive Order and Privacy Preserving Function family. To prevent the attackers from eavesdropping secret keys and pretending to be legal data users submitting searches, we propose a novel dynamic secret key generation protocol and a new data user authentication protocol. Furthermore, PRMSM supports efficient data user revocation. Extensive experiments on real-world datasets confirm the efficacy and efficiency of PRMSM.

Index Terms—Cloud computing, ranked keyword search, multiple owners, privacy preserving, dynamic secret key

1 Introduction

Cloud computing is a subversive technology that is changing the way IT hardware and software are designed and purchased [1]. As a new model of computing, cloud computing provides abundant benefits including easy access, decreased costs, quick deployment and flexible resource management, etc. Enterprises of all sizes can leverage the cloud to increase innovation and collaboration.

Despite the abundant benefits of cloud computing, for privacy concerns, individuals and enterprise users are reluctant to outsource their sensitive data, including emails, personal health records and government confidential files, to the cloud. This is because once sensitive data are outsourced to a remote cloud, the corresponding data owners lose direct control of these data [2]. Cloud service providers (CSPs) would promise to ensure owners' data security using mechanisms like virtualization and firewalls. However, these mechanisms do not protect owners' data privacy from the CSP itself, since the CSP possesses full control of cloud hardware, software, and owners' data. Encryption on sensitive data before outsourcing can preserve data privacy against CSP. However, data encryption makes the traditional data utilization service based on plaintext keyword search a very challenging problem. A trivial solution to this problem is to download all the encrypted data and decrypt them locally. However, this method is obviously impractical because it will cause a huge amount of communication overhead. Therefore, developing a secure search service over encrypted cloud data is of paramount importance.

Secure search over encrypted data has recently attracted the interest of many researchers. Song et al. [3] first define and solve the problem of secure search over encrypted data. They propose the conception of searchable encryption, which is a cryptographic primitive that enables users to perform a keyword-based search on an encrypted dataset, just as on a plaintext dataset. Searchable encryption is further developed by [4], [5], [6], [7], [8]. However, these schemes are concerned mostly with single or boolean keyword search. Extending these techniques for ranked multi-keyword search will incur heavy computation and storage costs. Secure search over encrypted cloud data is first defined by Wang et al. [9] and further developed by [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]. These researches not only reduce the computation and storage cost for secure keyword search over encrypted cloud data, but also enrich the

- *W. Zhang, Y.p. Lin, S. Xiao, and S.w. Zhou are with College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China, and with Hunan Provincial Key Laboratory of Dependable Systems and Networks, Changsha, 410082, China. Yaping Lin is the Corresponding Author. E-mail: {zhangweidoc, yplin, xiaosheng, swzhou}@hnu.edu.cn.*
- *J. Wu is with Department of Computer and Information Sciences, Temple University, 1805 N. Broad St., Philadelphia, PA 19122. E-mail: jiewu@temple.edu.*

category of search function, including secure ranked multi-keyword search, fuzzy keyword search, and similarity search. However, all these schemes are limited to the single-owner model. As a matter of fact, most cloud servers in practice do not just serve one data owner; instead, they often support multiple data owners to share the benefits brought by cloud computing. For example, to assist the government in making a satisfactory policies on health care service, or to help medical institutions conduct useful research, some volunteer patients would agree to share their health data on the cloud. To preserve their privacy, they will encrypt their own health data with their secret keys. In this scenario, only the authorized organizations can perform a secure search over this encrypted data contributed by multiple data owners. Such a Personal Health Record sharing system, where multiple data owners are involved, can be found at mymedwall.com.

Compared with the single-owner scheme, developing a full-fledged multi-owner scheme will have many new challenging problems. First, in the single-owner scheme, the data owner has to stay online to generate trapdoors (encrypted keywords) for data users. However, when a huge amount of data owners are involved, asking them to stay online simultaneously to generate trapdoors would seriously affect the flexibility and usability of the search system. Second, since none of us would be willing to share our secret keys with others, different data owners would prefer to use their own secret keys to encrypt their secret data. Consequently, it is very challenging to perform a secure, convenient, and efficient search over the data encrypted with different secret keys. Third, when multiple data owners are involved, we should ensure efficient user enrollment and revocation mechanisms, so that our system enjoys excellent security and scalability.

In this paper, we propose PRMSM, a privacy preserving ranked multi-keyword search protocol in a multi-owner cloud model. To enable cloud servers to perform secure search without knowing the actual value of both keywords and trapdoors, we systematically construct a novel secure search protocol. As a result, different data owners use different keys to encrypt their files and keywords. Authenticated data users can issue a query without knowing secret keys of these different data owners. To rank the search results and preserve the privacy of relevance scores between keywords and files, we propose a new additive order and privacy preserving function family, which helps the cloud server return the most relevant search results to data users without revealing any sensitive information. To prevent the attackers from eavesdropping secret keys and pretending to be legal data users submitting searches, we propose a novel dynamic secret key generation protocol and a new data user authentication protocol. As a result, attackers who steal the secret key and perform illegal searches would be easily detected. Furthermore, when we want to revoke a data user, PRMSM ensures efficient data user revocation. Extensive experiments on real-world datasets confirm the efficacy and efficiency of our proposed schemes.

The main contributions of this paper are listed as follows:

- We define a multi-owner model for privacy preserving keyword search over encrypted cloud data.
- We propose an efficient data user authentication protocol, which not only prevents attackers from eavesdropping secret keys and pretending to be illegal data users performing searches, but also enables data user authentication and revocation.
- We systematically construct a novel secure search protocol, which not only enables the cloud server to perform secure ranked keyword search without knowing the actual data of both keywords and trapdoors, but also allows data owners to encrypt keywords with self-chosen keys and allows authenticated data users to query without knowing these keys.
- We propose an Additive Order and Privacy Preserving Function family (AOPPF) which allows data owners to protect the privacy of relevance scores using different functions according to their preference, while still permitting the cloud server to rank the data files accurately.
- We conduct extensive experiments on real-world datasets to confirm the efficacy and efficiency of our proposed schemes.

The rest of this paper is organized as follows. Section 2 formulates the problem. Section 3 presents the preliminaries. Section 4 demonstrates how to perform user authentication. Section 5 introduces our novel secure search protocol. Section 6 defines AOPPF and illustrates how to use this technique to perform privacy-preserving ranked search. Section 7 presents security analysis. Section 8 demonstrates the efficiency of our proposed scheme. The related works are reviewed in Section 9. In Section 10, we conclude the paper.

2 Problem Formulation

In this section, we present a formal description for the target problem in this paper. We first define a system model and a corresponding threat model. Then we elucidate the design goals of our solution scheme and a list of notations used in later discussions.

2.1 System Model

In our multi-owner and multi-user cloud computing model, four entities are involved, as illustrated in Fig. 1; they are data owners, the cloud server,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2448099, IEEE Transactions on Computers

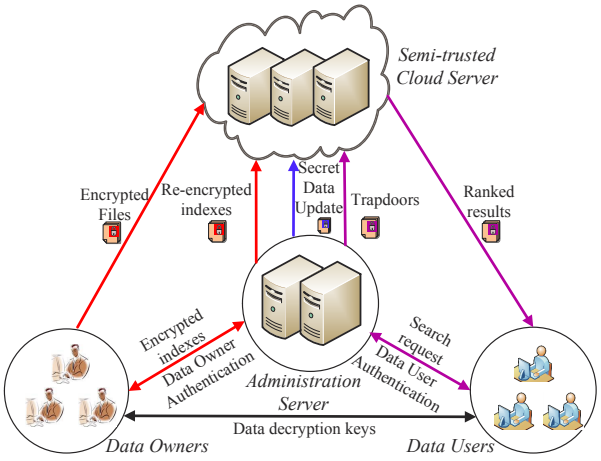


Fig. 1: Architecture of privacy preserving keyword search in a multi-owner and multi-user cloud model

administration server, and data users. Data owners have a collection of files $\mathcal{F}$. To enable efficient search operations on these files which will be encrypted, data owners first build a secure searchable index $\mathcal{I}$ on the keyword set $\mathcal{W}$ extracted from $\mathcal{F}$, then they submit $\mathcal{I}$ to the administration server. Finally, data owners encrypt their files $\mathcal{F}$ and outsource the corresponding encrypted files $\mathcal{C}$ to the cloud server. Upon receiving $\mathcal{I}$, the administration server re-encrypts $\mathcal{I}$ for the authenticated data owners and outsources the re-encrypted index to the cloud server. Once a data user wants to search t keywords over these encrypted files stored on the cloud server, he first computes the corresponding trapdoors and submits them to the administration server. Once the data user is authenticated by the administration server, the administration server will further re-encrypt the trapdoors and submit them to the cloud server. Upon receiving the trapdoor $\mathcal{T}$, the cloud server searches the encrypted index $\mathcal{I}$ of each data owner and returns the corresponding set of encrypted files. To improve the file retrieval accuracy and save communication cost, a data user would tell the cloud server a parameter k and cloud server would return the top-k relevant files to the data user. Once the data user receives the top-k encrypted files from the cloud server, he will decrypt these returned files. Note that how to achieve decryption capabilities are out of the scope of this paper; some excellent work regarding this problem can be found in [23].

2.2 Threat Model

In our threat model, we assume the administration server is trusted. The administrative server can be any trusted third party, e.g., the Certificate Authority in the Public Key Infrastructure, the aggregation and distribution layer in [24], and the third party auditor in [2]. Data owners and data users who passed the authentication of the administration server are also trusted. However, the cloud server is not trusted. Instead, we treat the cloud server as 'curious but honest' which is the same as in previous works [9], [10], [11], [13], [19]. The cloud server follows our proposed protocol, but it is eager to obtain the contents of encrypted files, keywords, and relevance scores. Note that preserving the access pattern, i.e., the list of returned files, is extremely expensive since the algorithm has to 'touch' the whole file set [25]. We do not aim to protect it in this work for efficiency concerns.

2.3 Design Goals and Security Definitions

To enable privacy preserving ranked multi-keyword search in the multi-owner and multi-user cloud environment, our system design should simultaneously satisfy security and performance goals.

- **Ranked Multi-keyword Search over Multi-owner:** The proposed scheme should allow multi-keyword search over encrypted files which would be encrypted with different keys for different data owners. It also needs to allow the cloud server to rank the search results among different data owners and return the top-k results.
- **Data owner scalability:** The proposed scheme should allow new data owners to enter this system without affecting other data owners or data users, i.e., the scheme should support data owner scalability in a plug-and-play model.
- **Data user revocation:** The proposed scheme should ensure that only authenticated data users can perform correct searches. Moreover, once a data user is revoked, he can no longer perform correct searches over the encrypted cloud data.
- **Security Goals:** The proposed scheme should achieve the following security goals: 1) Keyword Semantic Security (Definition 1). We will prove that PRMSM achieves semantic security against the chosen keyword attack. 2) Keyword secrecy (Definition 2). Since the adversary $\mathcal{A}$ can know whether an encrypted keyword matches a trapdoor, we use the weaker security goal (i.e., secrecy), that is, we should ensure that the probability for the adversary $\mathcal{A}$ to infer the actual value of a keyword is negligibly more than randomly guessing. 3) Relevance score secrecy. We should ensure that the cloud server cannot infer the actual value of the encoded relevance scores.

Definition 1: Given a probabilistic polynomial time adversary $\mathcal{A}$, he asks the challenger $\mathcal{B}$ for the cipher-text of his submitted keywords for polynomial times. Then $\mathcal{A}$ sends two keywords w_0 and w_1, which are not challenged before, to $\mathcal{B}$. $\mathcal{B}$ randomly sets $\mu \in \{0, 1\}$, and returns an encrypted keyword $\hat{w}_\mu$ to $\mathcal{A}$. $\mathcal{A}$ continues to ask $\mathcal{B}$ for the cipher-text of keyword w, the only restriction is that w is not w_0 or w_1. Finally, $\mathcal{A}$ outputs its guess μ' for μ. We define the advantage that $\mathcal{A}$

breaks PRMSM as $Adv_{\mathcal{A}}=\left|\Pr[\mu=\mu']-\frac{1}{2}\right|$. If $Adv_{\mathcal{A}}$ is negligible, we say that PRMSM is semantically secure against the chosen-keyword attack.

Definition 2: Given a probabilistic polynomial time adversary $\mathcal{A}$, he asks the challenger $\mathcal{B}$ for the ciphertext of his queried keywords for t times. Then $\mathcal{B}$ randomly chooses a keyword w^*, encrypts it to $\hat{w}^*$, and sends $\hat{w}^*$ to $\mathcal{A}$. $\mathcal{A}$ outputs its guess w' for w^*, and wins if $w'=w^*$. We define the probability that $\mathcal{A}$ breaks keyword secrecy as $Adv_{\mathcal{A}}=\Pr[w'=w^*]$. We say that PRMSM achieves keyword secrecy if $Adv_{\mathcal{A}}=\frac{1}{u-t}+\varepsilon$, where ϵ is a negligible parameter, t denotes the number of keywords that $\mathcal{A}$ has known, and u denotes the size of keyword dictionary.

2.4 Notations

- $\mathcal{O}$: the data owner collection, denoted as a set of m data owners $\mathcal{O}=(O_1,O_2,\ldots,O_m)$.
- $\mathcal{F}_i$: the plaintext file collection of O_i, denoted as a set of n data file $\mathcal{F}_i$=($F_{i,1}$, $F_{i,2}$,...,$F_{i,n}$).
- $\mathcal{C}_i$: the ciphertext file collection of $\mathcal{F}_i$, denoted as $\mathcal{C}_i$=($C_{i,1}$, $C_{i,2}$,...,$C_{i,n}$).
- $\mathcal{W}$: the keyword collection, denoted as a set of u keywords $\mathcal{W}=(w_1,w_2,\ldots,w_u)$.
- $\widehat{W}_i$: O_i's encrypted keyword collection of $\mathcal{W}$, denoted as $\widehat{W}_i=(\widehat{w}_{i,1},\widehat{w}_{i,2},\ldots,\widehat{w}_{i,u})$.
- $\widetilde{\mathcal{W}}$: the subset of $\mathcal{W}$ which represents queried keywords, denoted as $\widetilde{\mathcal{W}}=(w_1,w_2,\ldots,w_q)$.
- $\mathcal{T}_{\widetilde{\mathcal{W}}}$: the trapdoor for $\widetilde{\mathcal{W}}$, denoted as $\mathcal{T}_{\widetilde{\mathcal{W}}}=(T_{w_1},T_{w_2},\ldots,T_{w_q})$.
- $\mathcal{S}_{i,j,t}$: the relevance score of tth keyword to jth file of ith data owner.

3 Preliminaries

Before we introduce our detailed construction, we first briefly introduce some techniques that will be used in this paper.

3.1 Bilinear Map

Let G and G_1 denote two cyclic groups with a prime order p. We further denote g and g_1 as the generator of G and G_1, respectively. Let $\hat{e}$ be a bilinear map $\hat{e}: G\times G\rightarrow G_1$, then the following three conditions are satisfied: 1) Bilinear: $\forall a,b\in\mathbb{Z}_p^*$, $\hat{e}(g^a,g^b)=\hat{e}(g,g)^{ab}$. 2) Non-degenerate: $\hat{e}(g,g)\neq 1$. 3) Computable: $\hat{e}$ can be efficiently computed.

3.2 Bilinear Diffie-Hellman Problem and Bilinear Diffie-Hellman Assumption

The Bilinear Diffie-Hellman (BDH) problem in $(G,G_1,\hat{e})$ is described as follows, given random $g\in G$, and g^a,g^b,g^c for some $a,b,c\in\mathbb{Z}_p^*$, compute $\hat{e}(g,g)^{abc}\in G_1$.

The BDH assumption is presented as follows, given $(G,G_1,\hat{e})$, $g\in G$, and g^a,g^b,g^c for some $a,b,c\in\mathbb{Z}_p^*$, an adversary $\mathcal{A}$ has advantage ϵ in solving BDH when $Pr[\mathcal{A}(g^a,g^b,g^c)=\hat{e}(g,g)^{abc}]\geq\epsilon$. The BDH assumption tells that the advantage ϵ is negligible for any polynomial time $\mathcal{A}$.

4 Data User Authentication

To prevent attackers from pretending to be legal data users performing searches and launching statistical attacks based on the search result, data users must be authenticated before the administration server re-encrypts trapdoors for data users. Traditional authentication methods often follow three steps. First, data requester and data authenticator share a secret key, say, k_0. Second, the requester encrypts his personally identifiable information d_0 using k_0 and sends the encrypted data $(d_0)_{k_0}$ to the authenticator. Third, the authenticator decrypts the received data with k_0 and authenticates the decrypted data. However, this method has two main drawbacks. First, since the secret key shared between the requester and the authenticator remains unchanged, it is easy to incur replay attack. Second, once the secret key is revealed to attackers, the authenticator cannot distinguish between the legal requester and the attackers; the attackers can pretend to be legal requesters without being detected.

In this section, we first give an overview of the data user authentication protocol. Then, we introduce how to achieve secure and efficient data user authentication. Finally, we demonstrate how to detect illegal searches and how to enable secure and efficient data user revocation.

4.1 Overview

Now we give an example to illustrate the main idea of the user authentication protocol(the detailed protocol is elaborated in the following subsections). Assume Alice wants to be authenticated by the administration server, so she starts a conversation with the server. The server then authenticates the contents of the conversation. If the contents are authenticated, both Alice and the server will generate the initial secret key according to the conversation contents. After the initialization, to be authenticated successfully, Alice has to provide the historical data of their conversations. If the authentication is successful, both Alice and the administration server will change their secret keys according the contents of the conversation. In this way, the secret keys keep changing dynamically; without knowing the correct historical data, an attacker cannot start a successful conversation with the administration server.

4.2 User Authentication

Before we introduce the dynamic key generation method and the authentication protocol, we first introduce the format of the authentication data. As shown

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2448099, IEEE Transactions on Computers

Request Counter	Last Request Time	Personally Identifiable Data	Random Number	CRC

Fig. 2: Format of Authentication Data

in Fig. 2, the authentication data consists of five parts. The request counter field records the number of search requests that the data user has submitted. The last request time field asks the data user to provide the historical data of his previous request time. The personally identifiable data (e.g., passport number, telephone number) field is used to identify a specific data user, while the random number and CRC field are further used to check whether the authentication data has been tampered with.

The key point of a successful authentication is to provide both the dynamically changing secret keys and the historical data of the corresponding data user. Let $k_{i,j}$ denotes the secret key shared between administration server and the jth data user U_j after i instances of search requests, and $d_{i,j}$ denotes the authentication data for the $(i+1)$th request of U_j. Our authentication protocol runs in the following six steps.

A. Data user U_j prepares his authentication data $d_{i,j}$, i.e., U_j needs to fill in all the fields of authentication data based on his historical data.

B. Data user U_j encrypts $d_{i,j}$ with the current secret key $k_{i,j}$ and submits the encrypted authentication data $(d_{i,j})_{k_{i,j}}$ to the administration server.

C. After submitting the authentication data, the data user U_j generates another secret key $k_{i+1,j} = k_{i,j} \oplus H(d_{i,j})$, and stores both $k_{i,j}$ and $k_{i+1,j}$.

D. Upon receiving U_j's encrypted authentication data, the administration server decrypts it with $k_{i,j}$.

E. The administration server checks the request counter, last request time, personally identifiable data and CRC, respectively. If the authentication succeeds, the administration server first generates a new secret key $k_{i+1,j} = k_{i,j} \oplus H(d_{i,j})$, then he replies a confirmation data $d_{i+1,j}$, and encrypts it with $k_{i+1,j}$. Otherwise, the administration server encrypts $d_{i+1,j}$ with secret key $k_{i,j}$.

F. Upon receiving a reply from the administration server, the data user U_j will try to decrypt it with $k_{i+1,j}$. If the decrypted data contains the confirmation data, the authentication is successful. Otherwise, the authentication is regarded as being unsuccessful. The data user deletes the new generated secret key $k_{i+1,j}$ and considers whether to start another authentication.

Fig. 3 shows an example of successful authentication between the administration server and the data user. As we can see, after each successful authentication process, the secret key will be changed dynamically according to the previous key and some historical data. Therefore, once an attacker steals a secret key, he can hardly get any benefits. On one hand, if the attacker knows nothing about the historical data of the legal data user, he cannot even construct a legal authentication data. On the other hand, if the legal data user performs another successful authentication, the previous secret key will be expired.

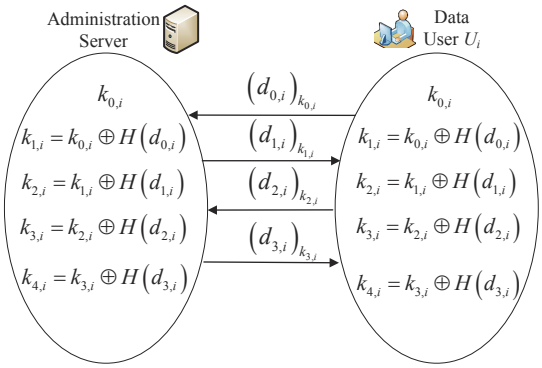


Fig. 3: Example of data user authentication and dynamic secret key generation

4.3 Illegal Search Detection

In our scheme, the authentication process is protected by the dynamic secret key and the historical information. We assume that an attacker has successfully eavesdropped the secret key $k_{0,j}$ of U_j. Then he has to construct the authentication data; if the attacker has not successfully eavesdropped the historical data, e.g., the request counter, the last request time, he cannot construct the correct authentication data. Therefore this illegal action will soon be detected by the administration server. Further, if the attacker has successfully eavesdropped all data of U_j, the attacker can correctly construct the authentication data and pretend himself to be U_j without being detected by the administration server. However, once the legal data user U_j performs his search, since the secret key on the administration server side has changed, there will be contradictory secret keys between the administration server and the legal data user. Therefore, the data user and administration server will soon detect this illegal action.

4.4 Data User Revocation

Different from previous works, data user revocation in our scheme does not need to re-encrypt and update large amounts of data stored on the cloud server. Instead, the adminstration server only needs to update the secret data S_a stored on the cloud server. As will be detailed in the next section, $S_a = g^{k_{a1} \cdot k_{a2} \cdot r_a}$, where k_{a1} and k_{a2} are the secret keys of the administration server, and r_a is randomly generated for every update operation. Consequently, the previous trapdoors will be expired. Additionally, without the help of the administration server, the revoked data user cannot generate the correct trapdoor $T_{w_{h'}}$. Therefore, a data user cannot perform correct searches once he is revoked.

5 Matching Different-key Encrypted Keywords

Numerous data owners are often involved in practical cloud applications. For privacy concerns, they would be reluctant to share secret keys with others. Instead, they prefer to use their own secret keys to encrypt their sensitive data (keywords, files). When keywords of different data owners are encrypted with different secret keys, the coming question is how to locate different-key encrypted keywords among multiple data owners. In this section, to enable secure, efficient and convenient searches over encrypted cloud data owned by multiple data owners, we systematically design schemes to achieve the following three requirements: First, different data owners use different secret keys to encrypt their keywords. Second, authenticated data users can generate their trapdoors without knowing these secret keys. Third, upon receiving trapdoors, the cloud server can find the corresponding keywords from different data owners' encrypted keywords without knowing the actual value of keywords or trapdoors.

5.1 Overview

Now we present an example to illustrate the main idea of our keywords matching protocol(the detailed protocol is elaborated in the following subsections). Assume Alice wants to use the cloud to store her file F, she first encrypts her file F, and gets the ciphertext C. To enable other users to perform secure searches on C, Alice extracts a keyword $w_{i,h}$, and sends the encrypted keyword $\hat{w}_{i,h} = (E_{a'}, E_o)$ to the administration server. The administration server further re-encrypts $E_{a'}$ to E_a, and submits $\hat{w}_{i,h} = (E_a, E_o)$ to the cloud server. Now Bob wants to search a keyword $w_{h'}$, he first generates the trapdoor ${T'}_{w_{h'}}$ and submits it to the administration server. The administration server re-encrypts ${T'}_{w_{h'}}$ to $T_{w_{h'}} = (T_1, T_2, T_3)$, generates a secret data S_a, and submits $T_{w_{h''}}$, S_a to the cloud server. The cloud server will judge whether Bob's search request matches Alice's encrypted keyword by checking whether $\hat{e}(E_a, T_3) = \hat{e}(E_o, T_1) \cdot \hat{e}(S_a, T_2)$ holds.

5.2 Construction Initialization

Our construction is based on the aforementioned bilinear map. Let g and g_1 denote the generator of two cyclic groups G and G_1 with order p. Let $\hat{e}$ be a bilinear map $\hat{e} : G \times G \to G_1$. Given different secret parameters as input, a randomized key generation algorithm will output the private keys used in the system. $k_{a1} \in \mathbb{Z}_p^+$, $k_{a2} \in \mathbb{Z}_p^+$, $k_{i,f} \in \mathbb{Z}_p^+$, $k_{i,w} \in \mathbb{Z}_p^+ \leftarrow (0,1)^*$, where k_{a1} and k_{a2} are the private keys of the administration server, $k_{i,w}$ and $k_{i,f}$ are the private keys used to encrypt keywords and files of data owner O_i, respectively. Let $H(\cdot)$ be a public hash function, its output locates in $\mathbb{Z}_p^+$.

5.3 Keyword Encryption

For keyword encryption, the following conditions should be satisfied: first, different data owners use their own secret keys to encrypt keywords. Second, for the same keyword, it would be encrypted to different cipher-texts each time. These properties benefit our scheme for two reasons. First, losing the key of one data owner would not lead to the disclosure of other owners' data. Second, the cloud server cannot see any relationship among encrypted keywords. Given the hth keyword of data owner O_i, i.e., $w_{i,h}$, we encrypt $w_{i,h}$ as follows.

$$\hat{w}_{i,h} = \left(g^{k_{i,w} \cdot r_o \cdot H(w_{i,h})}, g^{k_{i,w} \cdot r_o}\right) \quad (1)$$

where r_o is a randomly generated number each time, which helps enhance the security of $\hat{w}_{i,h}$. For easy description and understanding, we let $E'_a = g^{k_{i,w} \cdot r_o \cdot H(w_{i,h})}$ and $E_o = g^{k_{i,w} \cdot r_o}$.

The data owner delivers $E_{a'}$ and E_o to the administration server, and the administration server further re-encrypts $E_{a'}$ with his secret keys k_{a1} and k_{a2} and gets E_a.

$$E_a = \left(E_{a'} \cdot g^{k_{a1}}\right)^{k_{a2}} \quad (2)$$

Therefore $\hat{w}_{i,h} = (E_a, E_o)$. The administrative server further submits $\hat{w}_{i,h}$ to the cloud server. Note that, since the administration server only does simple computations on the encrypted data, he cannot learn any sensitive information from these random encrypted data without knowing the secret keys of data owners.

5.4 Trapdoor Generation

To make the data users generate trapdoors securely, conveniently and efficiently, our proposed scheme should satisfy two main conditions. First, the data user does not need to ask a large amount of data owners for secret keys to generate trapdoors. Second, for the same keyword, the trapdoor generated each time should be different. To meet this condition, the trapdoor generation is conducted in two steps: First, the data user generates trapdoors based on his search keyword and a random number. Second, the administration server re-encrypts the trapdoors for the authenticated data user.

Assume a data user wants to search keyword $w_{h'}$, so he encrypts it as follows:

$${T'}_{w_{h'}} = \left(g^{H(w_{h'}) \cdot r_u}, g^{r_u}\right) \quad (3)$$

where r_u is a randomly generated number each time. As we can see, during the trapdoor generation process, secret keys of data owners are not required. Additionally, with the help of random variable r_u, for the same keyword $w_{h'}$, we can generate two different trapdoors which prevent attackers from knowing the relationship among trapdoors.

Upon receiving $T'_{w_{h'}}$, the administration server first generates a random number r_a, and then re-encrypts $T'_{w_{h'}}$ as follows:

$$T_{w_{h'}} = \left(g^{H(w_{h'})\cdot r_u\cdot k_{a1}\cdot k_{a2}\cdot r_a}, g^{r_u\cdot k_{a1}}, g^{r_u\cdot k_{a1}\cdot r_a}\right) \quad (4)$$

For easy description and understanding, we let $T_1 = g^{H(w_{h'})\cdot r_u\cdot k_{a1}\cdot k_{a2}\cdot r_a}$, $T_2 = g^{r_u\cdot k_{a1}}$, $T_3 = g^{r_u\cdot k_{a1}\cdot r_a}$, hence, $T_{w_{h'}} = (T_1, T_2, T_3)$. Finally, the administration server submits $T_{w_{h'}}$ to the cloud server.

5.5 Keywords Matching among Different Data Owners

The cloud server stores all encrypted files and keywords of different data owners. The administration server will also store a secret data $S_a = g^{k_{a1}\cdot k_{a2}\cdot r_a}$ on the cloud server. Upon receiving a query request, the cloud will search over the data of all these data owners. The cloud processes the search request in two steps. First, the cloud matches the queried keywords from all keywords stored on it, and it gets a candidate file set. Second, the cloud ranks files in the candidate file set and finds the most top-k relevant files. We introduce the matching strategy here, while leaving the task of introducing the ranking strategy in the next section. When the cloud obtains the trapdoor $T_{w_{h'}}$ and encrypted keywords (E_o, E_a), he first computes

$$\begin{aligned} & \hat{e}\left(S_a, T_2\right) \\ = \ & \hat{e}\left(g^{r_a\cdot k_{a1}\cdot k_{a2}}, g^{r_u\cdot k_{a1}}\right) \\ = \ & \hat{e}(g,g)^{r_a\cdot k_{a1}\cdot k_{a2}\cdot r_u\cdot k_{a1}} \end{aligned} \quad (5)$$

Then he can judge whether $w_{h'} = w_{i,h}$ (i.e., an encrypted keyword is located) holds if the following equation is true.

$$\begin{aligned} & \hat{e}\left(E_a, T_3\right) \\ = \ & \hat{e}\left(\left(g^{k_{i,w}\cdot r_o\cdot H(w_{i,h})}\cdot g^{k_{a1}}\right)^{k_{a2}}, g^{r_u\cdot k_{a1}\cdot r_a}\right) \\ = \ & \hat{e}(g,g)^{(k_{i,w}\cdot r_o\cdot H(w_{i,h})+k_{a1})\cdot k_{a2}\cdot r_u\cdot k_{a1}\cdot r_a} \\ = \ & \hat{e}(g,g)^{k_{i,w}\cdot r_o\cdot H(w_{i,h})\cdot k_{a2}\cdot r_u\cdot k_{a1}\cdot r_a}\cdot \hat{e}\left(S_a, T_2\right) \\ = \ & \hat{e}\left(g^{k_{i,w}\cdot r_o}, g^{H(w_{i,h})\cdot k_{a2}\cdot r_u\cdot k_{a1}\cdot r_a}\right)\cdot \hat{e}\left(S_a, T_2\right) \\ = \ & \hat{e}\left(E_o, T_1\right)\cdot \hat{e}\left(S_a, T_2\right) \end{aligned} \quad (6)$$

6 Privacy Preserving Ranked Search

The aforementioned section helps the cloud match the queried keywords, and obtain a candidate file set. However, we cannot simply return undifferential files to data users for the following two reasons. First, returning all candidate files would cause abundant communication overhead for the whole system. Second, data users would only concern the top-k relevant files corresponding to their queries. In this section, we first elucidate an order and privacy preserving encoding scheme. Then we illustrate an additive order preserving and privacy preserving encoding scheme. Finally, we apply the proposed scheme to encode the relevance scores and obtain the top-k search results.

x	1	2	3	4	5
$f(x)$	100-1000	1100-1800	2000-4200	4300-5000	5100-7000

Fig. 4: An example of Order Preserving and Privacy Preserving Function

6.1 Order and Privacy Preserving Function

To rank the relevance score while preserving its privacy, the proposed function should satisfy the following conditions. 1) This function should preserve the order of data, as this helps the cloud server determine which file is more relevant to a certain keyword, according to the encoded relevance scores. 2) This function should not be revealed by the cloud server so that cloud server can make comparisons on encoded relevance scores without knowing their actual values. 3) Different data owners should have different functions such that revealing the encoded value of a data owner would not lead to the leakage of encoded values of other data owners. In order to satisfy condition 1, we introduce a data processing part $m(x,\cdot)$, which preserves the order of x. To satisfy condition 2, we introduce a disturbing part r_f which helps prevent the cloud server from revealing this function. To satisfy condition 3, we use $m(x,\cdot)$ to process the ID of data owners. So this function belongs to the following function family:

$$F^y_{oppf}(x) = \sum_{0\le j,k\le\tau} A_{j,k}\cdot m(x,j)\cdot m(y,k) + r_f \quad (7)$$

where τ denotes the degree of $F^y_{oppf}(x)$ and $A_{j,k}$ denotes the coefficients of $m(x,j)\cdot m(y,k)$. We further define $m(x,j)$ as follows: $m(x,0) = 1$, $m(x,1) = x$, and $m(x,j) = (m(x,j-1)+\alpha\cdot x)\cdot(1+\lambda)$ if $j>1$, where α and λ are two constant numbers.

Now we introduce how to set the disturbing part r_f. Since $\sum_{0\le j,k\le\tau} A_{j,k}\cdot(m(x+1,j)-m(x,j))\cdot m(y,k) \ge \sum_{0\le j,k\le\tau} A_{j,k}\cdot((1+\lambda)^{j-1} + \alpha\cdot\sum_{1\le i\le j-2}(1+\lambda))$.
Let l be an integer such that $2^{l-1}\le\sum_{0\le j,k\le\tau} A_{j,k}\cdot((1+\lambda)^{j-1}+\alpha\cdot\sum_{1\le i\le j-2}(1+\lambda))\le 2^l$, we can set $r_f \in (0, 2^{l-1})$.

Obviously, $\forall x_1>x_2$, we have $F^y_{oppf}(x_1)>F^y_{oppf}(x_2)$.

6.2 Additive Order and Privacy Preserving Function

With the order and privacy preserving function, we have: if $x_1 > x_2$, then $y_1 > y_2$. However, the addition of the function is not necessarily order preserving.

An example is presented in Fig. 4. Obviously, $f(x)$ is order preserving; to preserve privacy, a disturbing part is introduced. As we can see, $f(1) + f(5)$ varies from 5200 to 8000, $f(1) + f(4)$ varies from 4400 to 6000, $f(2) + f(3)$ varies from 3100 to 6000. Though

$1+5 > 1+4$ and $1+5 > 2+3$, it is probable for $f(1)+f(5) < f(1)+f(4)$ and $f(1)+f(5) < f(2)+f(3)$.

To correctly perform the secure ranked multi-keyword search, the sum of any two encoded relevance scores should still be ordered and privacy preserved (we use the sum of encoded relevance score to evaluate the relevance between a file and multiple keywords in this paper). To satisfy this condition, we further design an additive order and privacy preserving function family based on Eq. 7:

$$F^{y}_{aoppf}(x) = \sum_{0\le j,k\le \tau} A_{j,k}\cdot m(x,j)\cdot m(y,k) + r_{aof} \quad (8)$$

Where τ denotes the degree of $F^{y}_{aoppf}(x)$ and $A_{j,k}$ denotes the coefficients of $m(x,j)\cdot m(y,k)$.

Before we take a step further towards the use of $F^{y}_{aoppf}(x)$, we will first introduce our auxiliary theorem. i.e., how to make an order and privacy preserving function to be additive order and privacy preserving.

Definition 3: Given an order preserving function $y = f(x)$, we define $\Delta f(x_i) = f(x_{i+1}) - f(x_i)$ and $\tilde{\Delta} f(x_i) = \Delta f(x_{i+1}) - \Delta f(x_i)$. To preserve privacy, we define $y_i = f(x_i) + r_i$, where r_i is a random number and $r_i > 0$. We define the max value of r_i as r_{imax}, and $r_{imax} = f(x_i) - i\cdot\Delta(x_i)$, hence y_i can change from $f(x_i)$ to $f(x_i) + r_{imax}$.

Theorem 1: Given an order preserving function $y_i = f(x_i) + r_i$, when the following three conditions are satisfied: (1) $\forall i$, $\Delta f(x_i+1) \le \Delta f(x_i)$; (2) $\forall i$, $\left|\tilde{\Delta} f(x_i+1)\right| \ge \left|\tilde{\Delta} f(x_i)\right|$; (3) $\forall i$, $r_i < \left(i^2\cdot\tilde{\Delta} f(x_i)\right)/2$. The order and privacy preserving function y_i is also an additive order and privacy preserving, that is, for any $\sum_{x_i\in D_1} x_i \le \sum_{x_j\in D_2} x_j$, where D_1 and D_2 denotes two subsets of the definition domain, we have $\sum_{x_i\in D_1} y_i \le \sum_{x_j\in D_2} y_j$.

Proof: The proof is elaborated in Appendix A.

6.3 Encoding relevance scores

In our paper, we use a well-known method to compute the relevance score [26], i.e., $Score(w, F_d) = \frac{1}{|F_d|}(1+\ln f_{d,w})\ln(1+\frac{N}{f_w})$, where w denotes the given keyword, $|F_d|$ is the length of file F_d, $f_{d,w}$ denotes the TF of w in file F_d, f_w denotes the number of files containing w, and N denotes the total number of files in the collection.

Now we introduce how to encode the relevance score with an additive order and privacy preserving function in our additive order and privacy preserving function family $F^{y}_{aoppf}(x)$. Given input $\mathcal{S}_{i,j,t}$ (the relevance score of tth keyword to jth file of ith data owner). Data owners can independently choose a function from this family to protect the privacy of their relevance scores. For simplicity, in this paper, we specify data owner i to choose $F^{H_i(i)}_{aoppf}(\cdot)$ and define $\mathcal{V}_{i,j,t}$ as the encoded data of $\mathcal{S}_{i,j,t}$:

$$\mathcal{V}_{i,j,t} = F^{H_i(i)}_{aoppf}(\mathcal{S}_{i,j,t}) \quad (9)$$

where $H_i(\cdot)$ is a public known hash function, and i is the ID of data owner O_i.

With the well-designed properties of F_{aoppf}, the cloud server can make a comparison among encoded relevance scores for the same data owner. However, since different data owners encode their relevance scores with different functions in F_{aoppf}, the cloud server cannot make a comparison between encoded relevance scores for different data owners. To solve this problem, we define:

$$\mathcal{T}_{i,j,t}(y) = F^{y}_{aoppf}(\mathcal{S}_{i,j,t}) \quad (10)$$

where $\mathcal{T}_{i,j,t}(y)$ is used to help the cloud server make comparisons among relevance scores encoded by different data owners, and y is a variable which takes the hash value of data owner's ID as input.

Finally, we attach each $\mathcal{V}_{i,j,t}$ with a $\mathcal{T}_{i,j,t}(y)$.

6.4 Ranking search results

In this paper, we use the sum of the relevance scores as the metric to rank search results. Now, we introduce the strategies of ranking search results based on the encoded relevance scores. First, the cloud computes $\mathcal{V}_{i,j} = \sum_{t\in\widetilde{\mathcal{W}}} \mathcal{V}_{i,j,t}$, the sum of encoded relevance scores between the jth file and matched keywords for O_i, and the auxiliary value $\mathcal{T}_{i,j}(y) = \sum_{t\in\widetilde{\mathcal{W}}} \mathcal{T}_{i,j,t}(y)$. Then the cloud ranks the sum of encoded relevance score with the following two conditions:

(1) Two encoded data belong to the same data owner. Given that a data user issues a query $\widetilde{\mathcal{W}} = \{w_m, w_n\}$, we assume that O_i's F_1 and F_2 satisfy the query. Then the cloud adds the encoded relevance score together and gets the relevance score of $O_i's\ F_1$ to $\widetilde{\mathcal{W}}$:

$$\mathcal{V}_{i,1} = \mathcal{V}_{i,1,m} + \mathcal{V}_{i,1,n} \quad (11)$$

Similarly,

$$\mathcal{V}_{i,2} = \mathcal{V}_{i,2,m} + \mathcal{V}_{i,2,n} \quad (12)$$

If $\mathcal{V}_{i,1} \ge \mathcal{V}_{i,2}$, then F_1 is more relevant to the queried keyword sets $\widetilde{\mathcal{W}}$, otherwise, F_2 is more relevant to $\widetilde{\mathcal{W}}$.

(2) Two encoded data belong to two different data owners. Given that a data user issues a query $\widetilde{\mathcal{W}} = \{w_m, w_n\}$, we assume O_i's F_1 and O_j's F_2 satisfy the query. The cloud server makes comparison on their encoded relevance scores in the following four steps:

First, the cloud server computes the relevance score of $O_i's\ F_1$ to $\widetilde{\mathcal{W}}$

$$\mathcal{V}_{i,1} = \mathcal{V}_{i,1,m} + \mathcal{V}_{i,1,n} \quad (13)$$

Second, the cloud server computes the $\mathcal{T}_{j,2}(y)$

$$\mathcal{T}_{j,2}(y) = \mathcal{T}_{j,2,m}(y) + \mathcal{T}_{j,2,n}(y) \quad (14)$$

Third, the cloud server substitutes $H_i(i)$ for the variable y and gets $\mathcal{T}_{j,2}(H(i))$.

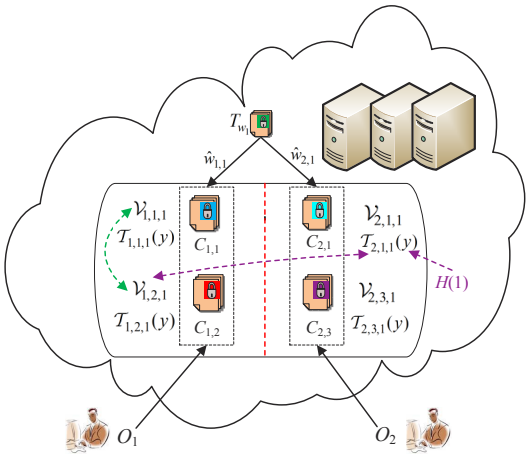


Fig. 5: Example of ranking search results

Finally, the cloud draws the conclusion: if $\mathcal{V}_{i,1} \geq \mathcal{T}_{j,2}(H_i(i))$, then $O_i's\ F_1$ is more relevant to $\widetilde{\mathcal{W}}$ than $O_j's\ F_2$, otherwise, $O_j's\ F_2$ is more relevant to $\widetilde{\mathcal{W}}$.

Now, the cloud server can make comparisons on encoded relevance scores. Thus, it is easy for the cloud to return the top-k relevant files to the data user.

Fig. 5 shows an example of making comparisons on the encoded relevance scores. As we can see, given the trapdoor T_{w_1}, the cloud server finds that $\hat{w}_{1,1}$ and $\hat{w}_{2,1}$ match T_{w_1}. Data owner O_1 has two encrypted files $C_{1,1}$ and $C_{1,2}$ that contain $\hat{w}_{1,1}$. Data owner O_2 also has two encrypted files $C_{2,1}$ and $C_{2,3}$ that contain $\hat{w}_{2,1}$. For O_1, he compares $\mathcal{V}_{1,1,1}$ with $\mathcal{V}_{1,2,1}$, if $\mathcal{V}_{1,1,1} > \mathcal{V}_{1,2,1}$, the cloud server regards $C_{1,1}$ as being more relevant to T_{w_1}; otherwise, $C_{2,1}$ is more relevant to T_{w_1}. We assume $C_{1,2}$ is more relevant to T_{w_1} for O_1 and $C_{2,1}$ is more relevant for O_2; then the cloud makes comparisons between $C_{1,2}$ and $C_{2,1}$. As shown in the figure, the cloud first substitutes $H(1)$ to $\mathcal{T}_{2,1,1}(y)$ and gets $\mathcal{T}_{2,1,1}(H(1))$, then he compares $\mathcal{T}_{2,1,1}(H(1))$ with $\mathcal{V}_{1,2,1}$, if $\mathcal{V}_{1,2,1} > \mathcal{T}_{2,1,1}(H(1))$. Then O_1's encrypted file $C_{1,2}$ is more relevant to T_{w_1}, otherwise, O_2's encrypted file $C_{2,1}$ is more relevant to the search request.

7 Security Analysis

In this section, we provide step-by-step security analyses to demonstrate that the security requirements have been satisfied for the data files, the keywords, the queries, and the relevance scores.

7.1 Data Files

The data files are protected by symmetric encryption before upload. As long as the encryption algorithm is not breakable, the cloud server cannot know the data.

7.2 Keywords

We formulate the security goals achieved by PRMSM with the following two theorems.

Theorem 2: Given the DDH (Decisional DiffieHellman) assumption, PRMSM is semantically secure against the chosen keyword attack under the selective security model.

Proof: See Appendix B.

Theorem 3: Given the DL (Discrete Logarithm) assumption, PRMSM achieves keyword secrecy in the random oracle model.

Proof: See Appendix C.

7.3 Trapdoors

Recall the trapdoor construction formula,

$$T_{w_{h'}} = \left(g^{H(w_{h'})\cdot r_u \cdot k_{a1} \cdot k_{a2} \cdot r_a}, g^{r_u \cdot k_{a1}}, g^{r_u \cdot k_{a1} \cdot r_a}\right) \quad (15)$$

If the cloud server wants to know the actual value of the trapdoor, and distinguish two trapdoors, it has to solve the discrete logarithm problem in $\mathbb{Z}_p$ with large prime p, therefore, the privacy of trapdoor is protected as long as the discrete logarithm problem is hard.

7.4 Relevance Scores

In our scheme, relevance scores are encoded with two Additive Order and Privacy Preserving Functions. Now we analyze the security of additive order and privacy preserving functions. Assume the input for the $F^y_{aopp}(x)$ is s, and the data owner ID is i. Then the cloud server can only capture the following value that is derived from s:

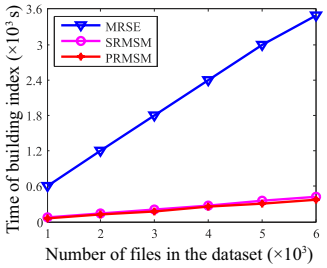
$$F^{(H_i(i))}_{aopp}(s) = \sum_{0 \leq j,k \leq \tau} A_{j,k} \cdot m(s,j) \cdot m(H_i(i),k) + r_{aof} \quad (16)$$

Assume the cloud server has collected n encoded relevance scores for the same relevance score s ($F^{(H_i(i))}_{aopp}(s)$), then he can construct n equations. However, these functions have $n+1$ unknown variables. Therefore, it is infeasible for the cloud server to break the additive order and privacy preserving $F^y_{aopp}(x)$. Thus, the security of the corresponding $F^y_{aopp}(x)$ encoded relevance score is also preserved.

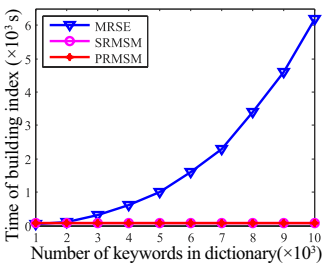
8 Performance Evaluation

In this section, we measure the efficiency of PRMSM, and compare it with its previous version, Secure Ranked Multi-keyword Search for Multiple data owners in cloud computing (SRMSM) [27], and the state-of-the-art, privacy-preserving Multi-keyword Ranked Search over Encrypted cloud data (MRSE) [11], side by side. Since MRSE is only suitable for the single owner model, our PRMSM and SRMSM not only work well in multi-owner settings, but also outperform MRSE on many aspects.

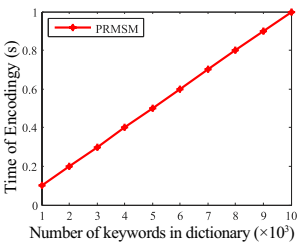
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2448099, IEEE Transactions on Computers



(a) For different size datasets with the same dictionary, u = 4000.

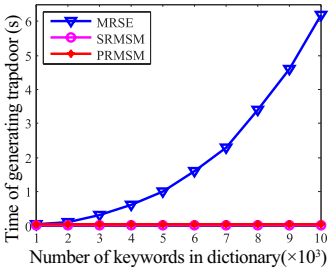


(b) For the same dataset with different dictionary sizes, n = 1000.

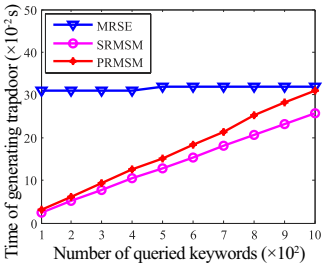


(c) Time cost of encoding relevance score (a subroutine of building index)

Fig. 6: Time cost of index construction.



(a) For different size of keyword dictionary with the same number of queried keywords, q = 100.



(b) For different number of queried keywords with the same size of keyword dictionary, u = 4000.

Fig. 7: Time cost of generating trapdoors.

8.1 Evaluation Settings

We conduct performance experiments on a real data set, the Internet Request For Comments dataset (RFC) [28]. We use Hermetic Word Frequency Counter [29] to extract keywords from each RFC file. After the keyword extraction, we compute keyword statistics such as the keyword frequency in each file, the length of each file, the number of files containing a specific keyword, etc. We further calculate the relevance score of a keyword to a file based on these statistics. The file size and keyword frequency of this data set can be seen in [20].

The experiment programs are coded using the Python programming language on a PC with 2.2GHZ Intel Core CPU and 2GB memory. We implement all necessary routines for data owners to preprocess data files: for the data user to generate trapdoors, for the administrative server to re-encrypt keywords, trapdoors, and for the cloud server to perform ranked searches. We use the Weil pairing [30] to construct our bilinear map.

8.2 Evaluation Results

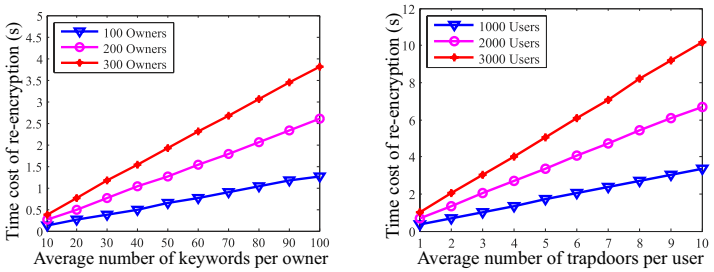
8.2.1 Index Construction

Fig. 6(a) shows that, given the same keyword dictionary (u=4000), time of index construction for these schemes increases linearly with an increasing number of files, while SRMSM and PRMSM spend much less time on index construction. Fig. 6(b) demonstrates that, given the same number of files (n=1000), SRMSM and PRMSM consume much less time than MRSE on constructing indexes. Additionally, SRMSM and PRMSM are insensitive to the size of the keyword dictionary for index construction, while MRSE suffers a quadratic growth with the size of keyword dictionary increases. Fig. 6(c) shows the encoding efficiency of our proposed AOPPF. The time spent on encoding increases from 0.1s to 1s when the number of keywords increases from 1000 to 10000. This time cost can be acceptable.

8.2.2 Trapdoor Generation

Compared with index construction, trapdoor generation consumes relatively less time. Fig. 7(a) demonstrates that, given the same number of queried keywords (q=100), SRMSM and PRMSM are insensitive to the size of keyword dictionary on trapdoor generation and consumes 0.026s and 0.031s, respectively. Meanwhile, MRSE increases from 0.04s to 6.2s. Fig. 7(b) shows that, given the same number of dictionary size (u=4000), when the number of queried keywords increases from 100 to 1000, the trapdoor generation time for MRSE is 0.31s, and remains unchanged. While SRMSM increases from 0.024s to 0.25s, PRMSM increases from 0.031s to 0.31s. We observe that PRMSM spends a little more time than SRMSM on trapdoor generation; the reason is that PRMSM introduces an additional variable to ensure the randomness of trapdoors.

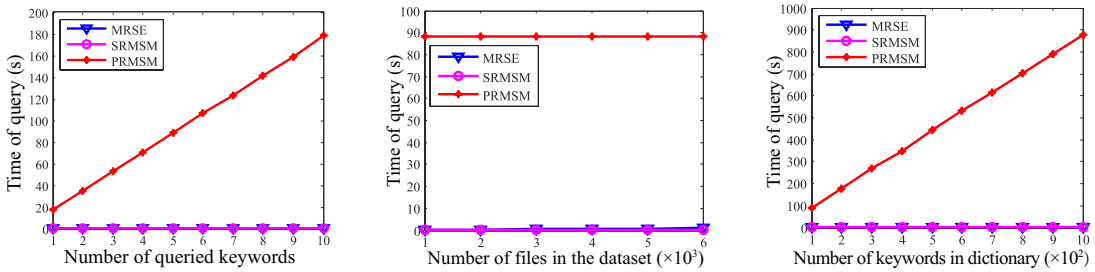
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2448099, IEEE Transactions on Computers



(a) Time cost of re-encrypting keywords

(b) Time cost of re-encrypting trapdoors.

Fig. 8: Time cost of the administration server.



(a) For different size of queried keywords with the same size of data set, n=2000 and keyword dictionary, u = 100.

(b) For different size of data set with the same size of queried keywords, q=5 and keyword dictionary, u = 100.

(c) For different size of keyword dictionary with the same size of queried keywords, q=5 and data set, n = 2000.

Fig. 9: Time cost of search.

8.2.3 Re-encryption by the administration server

Fig. 8(a) illustrates the re-encryption time cost of the administration server in PRMSM. As we can see, for the same average number of keywords per owner, the more data owners are involved, the more time is spent on re-encryption. When there are 300 data owners, each data owner has 100 keywords; we need 3.8s to re-encrypt these keywords, which is acceptable.

Fig. 8(b) demonstrates the time cost of re-encrypting trapdoors. We observe that, for the same average number of trapdoors per user, the more data users that submit trapdoors, the more time would be spent on re-encryption. When there are 1000 data users who concurrently submit data, each data user has 10 trapdoors; we only need 3.34s for re-encryption.

8.2.4 Search

From Fig. 9, we observe that, PRMSM spends more time for searching. The fundamental reason is that, the pairing operation used in PRMSM needs more time. As we can see from Fig. 9(a) and Fig. 9(c), the more keywords existing in the cloud server, the more time is required for pairing operation. Fig. 9(b) confirms that when the number of keywords stored on the cloud server remains a constant, PRMSM will not increase even if the number of files increases. Though PRMSM spends relatively more time, this observation also confirms that the searching operation should be outsourced to the cloud server.

9 Related Work

In this section, we review three categories of work: searchable encryption, secure keyword search in cloud computing, and order preserving encryption.

9.1 Searchable Encryption

The earliest attempt of searchable encryption was made by Song et al. In [3], they propose to encrypt each word in a file independently and allow the server to find whether a single queried keyword is contained in the file without knowing the exact word. This proposal is more of theoretic interests because of high computational costs. Goh et al. propose building a keyword index for each file and using Bloom filter to accelerate the search [4]. Curtmola et al. propose building indices for each keyword, and use hash tables as an alternative approach to searchable encryption [5]. The first public key scheme for keyword search over encrypted data is presented in [6]. [7] and [8] further enrich the search functionalities of searchable encryption by proposing schemes for conjunctive keyword search.

The searchable encryption cares mostly about single keyword search or boolean keyword search. Extending these techniques for ranked multi-keyword search will incur heavy computation and storage costs.

9.2 Secure Keyword Search in Cloud Computing

The privacy concerns in cloud computing motivate the study on secure keyword search. Wang et al.

first defined and solved the secure ranked keyword search over encrypted cloud data. In [9] and [18], they proposed a scheme that returns the top-k relevant files upon a single keyword search. Cao et al. [10], [11], and Sun et al. [31], [12] extended the secure keyword search for multi-keyword queries. Their approaches vectorize the list of keywords and apply matrix multiplications to hide the actual keyword information from the cloud server, while still allowing the server to find out the top-k relevant data files. Xu et al. proposed MKQE (Multi-Keyword ranked Query on Encrypted data) that enables a dynamic keyword dictionary and avoids the ranking order being distorted by several high frequency keywords [13]. Li et al. [14], Chuah et al. [15], Xu et al. [16] and Wang et al. [17] proposed fuzzy keyword search over encrypted cloud data aiming at tolerance of both minor typos and format inconsistencies for users' search input. [19] further proposed privacy-assured similarity search mechanisms over outsourced cloud data. In [20], we proposed a secure, efficient, and distributed keyword search protocol in the geo-distributed cloud environment.

The system model of these previous works only consider one data owner, which implies that in their solutions, the data owner and data users can easily communicate and exchange secret information. When numerous data owners are involved in the system, secret information exchanging will cause considerable communication overhead. Sun et al. [21] and Zheng et al. [22] proposed secure attribute-based keyword search schemes in the challenging scenario where multiple owners are involved. However, applying CP-ABE in the cloud system would introduce problems for data user revocation, i.e., the cloud has to update the large amount of data stored on it for a data user revocation [32]. Additionally, they do not support privacy preserving ranked multi-keyword search. Our paper differs from previous studies regarding the emphasis of multiple data owners in the system model. This paper seeks a solution scheme to maximally relax the requirements for data owners and users, so that the scheme could be suitable for a large number of cloud computing users.

9.3 Order Preserving Encryption

The order preserving encryption is used to prevent the cloud server from knowing the exact relevance scores of keywords to a data file. The early work of Agrawal et al. proposed an Order Preserving symmetric Encryption (OPE) scheme where the numerical order of plain texts are preserved [33]. Boldyreva et al. further introduced a modular order preserving encryption in [34]. Yi et al [35] proposed an order preserving function to encode data in sensor networks. Popa et al. [36] recently proposed an ideal-secure order-preserving encryption scheme. Kerschbaum et al. [37] further proposed a scheme which is not only idea-secure but is also an efficient order-preserving encryption scheme. However, these schemes are not additive order preserving. As a complementary work to the previous order preserving work, we propose a new additive order and privacy preserving functions (AOPPF). Data owners can freely choose any function from an AOPPF family to encode their relevance scores. The cloud server computes the sum of encoded relevance scores and ranks them based on the sum.

10 Conclusions

In this paper, we explore the problem of secure multi-keyword search for multiple data owners and multiple data users in the cloud computing environment. Different from prior works, our schemes enable authenticated data users to achieve secure, convenient, and efficient searches over multiple data owners' data. To efficiently authenticate data users and detect attackers who steal the secret key and perform illegal searches, we propose a novel dynamic secret key generation protocol and a new data user authentication protocol. To enable the cloud server to perform secure search among multiple owners' data encrypted with different secret keys, we systematically construct a novel secure search protocol. To rank the search results and preserve the privacy of relevance scores between keywords and files, we propose a novel Additive Order and Privacy Preserving Function family. Moreover, we show that our approach is computationally efficient, even for large data and keyword sets. As our future work, on one hand, we will consider the problem of secure fuzzy keyword search in a multi-owner paradigm. On the other hand, we plan to implement our scheme on the commercial clouds.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (Project No. 61173038, 61472125, 61300217), and the China Scholarship Council. A preliminary version of this paper appeared in the Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'14) [27].

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communication of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *Computers, IEEE Transactions on*, vol. 62, no. 2, pp. 362–375, 2013.

[3] D.Song, D.Wagner, and A.Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE International Symposium on Security and Privacy (S&P'00)*, Nagoya, Japan, Jan. 2000, pp. 44–55.

[4] E. Goh. (2003) Secure indexes. [Online]. Available: http://eprint.iacr.org/
[5] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proc. ACM CCS'06*, VA, USA, Oct. 2006, pp. 79–88.
[6] D. B. et al., "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *EUROCRYPT*, vol. 43, pp. 506–522, 2004.
[7] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. Applied Cryptography and Network Security (ACNS'04)*, Yellow Mountain, China, Jun. 2004, pp. 31–45.
[8] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. Information and Communications Security (ICICS'05)*, Beijing, China, Dec. 2005, pp. 414–426.
[9] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE Distributed Computing Systems (ICDCS'10)*, Genoa, Italy, Jun. 2010, pp. 253–262.
[10] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM'11*, Shanghai, China, Apr. 2011, pp. 829–837.
[11] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 222–233, 2014.
[12] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 11, pp. 3025–3035, 2014.
[13] Z. Xu, W. Kang, R. Li, K. Yow, and C. Xu, "Efficient multi-keyword ranked query on encrypted data in the cloud," in *Proc. IEEE Parallel and Distributed Systems (ICPADS'12)*, Singapore, Dec. 2012, pp. 244–251.
[14] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM'10*, San Diego, CA, Mar. 2010, pp. 1–5.
[15] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," in *Proc. IEEE 31th International Conference on Distributed Computing Systems (ICDCS'11)*, Minneapolis, MN, Jun. 2011, pp. 383–392.
[16] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *Computers, IEEE Transactions on*, vol. 62, no. 11, pp. 2266–2277, 2013.
[17] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *IEEE INFOCOM*, Toronto, Canada, May 2014, pp. 2112–2120.
[18] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 8, pp. 1467–1479, 2012.
[19] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proc. IEEE INFOCOM'12*, Orlando, FL, Mar. 2012, pp. 451–459.
[20] W. Zhang, Y. Lin, S. Xiao, Q. Liu, and T. Zhou, "Secure distributed keyword search in multiple clouds," in *Proc. IEEE/ACM IWQOS'14*, Hongkong, May 2014, pp. 370–379.
[21] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE INFOCOM'14*, Toronto, Canada, May 2014, pp. 226–234.
[22] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE INFOCOM'14*, Toronto, Canada, May 2014, pp. 522–530.
[23] T. Jung, X. Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *Proc. IEEE INFOCOM'13*, Turin, Italy, Apr. 2013, pp. 2625–2633.
[24] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Efficient information retrieval for ranked queries in cost-effective cloud environments," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2581–2585.
[25] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–981, 1998.
[26] I. H. Witten, A. Moffat, and T. C. Bell, *Managing gigabytes: Compressing and indexing documents and images*. San Francisco, USA: Morgan Kaufmann, 1999.
[27] W. Zhang, S. Xiao, Y. Lin, T. Zhou, and S. Zhou, "Secure ranked multi-keyword search for multiple data owners in cloud computing," in *Proc. 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN2014)*. Atlanta, USA: IEEE, jun 2014, pp. 276–286.
[28] IETF, "Request for comments database." [Online]. Available: http://www.ietf.org/rfc.html
[29] H. Systems, "Hermetic word frequency counter." [Online]. Available: http://www.hermetic.ch/wfc/wfc.htm
[30] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.
[31] 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. IEEE ASIACCS'13*, Hangzhou, China, May 2013, pp. 71–81.
[32] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 25, no. 10, pp. 2271–2282, 2013.
[33] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proc. ACM SIGMOD'04*, Paris, France, Jun. 2004, pp. 563–574.
[34] A. Boldyreva, N. Chenette, Y. Lee, and A. O, "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in *Proc. Advances in Cryptology (CRYPTO'11)*, California USA, Aug. 2011, pp. 578–595.
[35] Y. Yi, R. Li, F. Chen, A. X. Liu, and Y. Lin, "A digital watermarking approach to secure and precise range query processing in sensor networks," in *Proc. IEEE INFOCOM'13*, Turin, Italy, Apr. 2013, pp. 1950–1958.
[36] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 463–477.
[37] F. Kerschbaum and A. Schroepfer, "Optimal average-complexity ideal-security order-preserving encryption," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 275–286.

Wei Zhang received his B.S. degree in Computer Science from Hunan University, China, in 2011. Since 2011, He has been a Ph.D. candidate in College of Computer Science and Electronic Engineering, Hunan University. His research interests include cloud computing, network security and data mining.

Yaping Lin received his B.S. degree in Computer Application from Hunan University, China, in 1982, and his M.S. degree in Computer Application from National University of Defense Technology, China in 1985. He received his Ph.D. degree in Control Theory and Application from Hunan University in 2000. He has been a professor and Ph.D supervisor in Hunan University since 1996. From 2004-2005, he worked as a visiting researcher at the University of Texas at Arlington. His research interests include machine learning, network security and wireless sensor networks.

Sheng Xiao received his B.S. degree in Tsinghua University, China, in 2002 and his M.S. degree from the National Singapore University/MIT (Singapore-MIT Alliance) in 2003. He received his PhD degree from the University of Massachusetts, Amherst, in 2013. He has been an associate professor at the College of Computer Science and Electronic Engineering, Hunan University. He received INFOCOM Best Paper in 2010. His research interests include communication security, cloud computing and big data.

Jie Wu is the chair and a Laura H. Carnell professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA. Prior to joining Temple University, he was a program director with the National Science Foundation and a distinguished professor with Florida Atlantic University, Boca Raton, FL. He regularly publishes in scholarly journals, conference proceedings, and books. His research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He serves on several editorial boards, including IEEE Transactions on Computers, IEEE Transactions on Service Computing, and Journal of Parallel and Distributed Computing. He was general cochair/chair for IEEE MASS 2006, IEEE IPDPS 2008, and IEEE ICDCS 2013, as well as program cochair for IEEE INFOCOM 2011 and CCF CNCC 2013. he served as a general chair forACM MobiHoc 2014. He was an IEEE Computer Society distinguished visitor, ACM distinguished speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). He is a CCF distinguished speaker and a fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Siwang Zhou received his B.S. degree in Applied Physics from Fudan University in 1995. He received his PhD degree in computer application technology from Hunan University in 2007. He has been an associate professor at the College of Computer Science and Electronic Engineering, Hunan University. His research interests include wireless sensor networks and wavelet compressive sensing.