Future Generation Computer Systems 100 (2019) 689–700

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners

Hui Yin [a], Zheng Qin [b,*], Jixin Zhang [b], Lu Ou [b], Fangmin Li [a], Keqin Li [c]

[a] College of Computer Engineering and Applied Mathematics, Changsha University, Changsha, Hunan, 410022, China
[b] College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, 410082, China
[c] Department of Computer Science, State University of New York, New Paltz, NY, 12561, USA

## HIGHLIGHTS

- Presented a secure conjunctive multi-keyword query scheme for multiple data owners.
- Presented a secure query results ranking mechanism.
- Analyzed and proved the security of the proposed scheme.
- Analyzed the complexity and tested the performance of the proposed scheme.

## ARTICLE INFO

*Article history:*
Received 20 September 2017
Received in revised form 27 July 2018
Accepted 1 May 2019
Available online 24 May 2019

*Keywords:*
Cloud computing
Conjunctive keyword query
Multiple data owners
Query results ranking
Secure search

## ABSTRACT

Recently, secure search over encrypted cloud data has become a hot research spot and challenging task. A number of secure search schemes have been proposed to try to meet this challenge. However, most of them only consider the single data owner model. In this paper, we propose a conjunctive multi-keyword ranked secure search scheme for multiple data owners. To guarantee data security and system flexibility in the multiple data owners environment, we design an ingenious secure query scheme that allows each data owner to adopt randomly chosen temporary keys to build secure indexes for different data files. An authorized data user does not need to know these temporary keys of constructing indexes and can instead randomly choose another temporary query keys to encrypt query keywords, while the cloud server can correctly perform keywords matching over encrypted data files. To rank the query results of a conjunctive multi-keyword query, the cloud server computes the similarity scores between the query and its query results according to encrypted relevance scores of keywords without obtaining any sensitive information. Extensive experiments demonstrate the correctness and practicality of the proposed scheme.



## 1. Introduction

### 1.1. Motivation

We have witnessed the surge and remarkable growth of cloud computing technologies in the past few years, which have emerged as a new paradigm for computation, storage, and hosting services. As cloud computing technologies become mature, more and more enterprise and individual users are motivated to outsource their private data to the cloud server for enjoying the abundant benefits brought by the cloud computing, such as economic savings, great flexibility, quick deployment, elastic computation, and on-demand high quality services.

The production and development of any one of new things are a double-edged sword. Cloud computing is no exception, which plays a huge advantage compared with traditional information technologies, meanwhile, also brings many new problems and challenges. Among them, the data privacy and security issues have been paid much attention with the increasing cloud security incidents [1]. Data encryption is an effective way to protect the confidentiality of cloud data [2,3]. However, data encryption makes effective data retrieval and utilization a very challenging task, which greatly weakens abilities of data processing of cloud computing.

Searchable encryption is a booming cryptographic primitive that supports effective keyword search over encrypted data. As early as 2000, Song et al. had set out to explore the problem of search over encrypted data and first introduced a practical secure query technique in [4]. Later, searchable encryption was formally defined and further developed by [5–10]. Recently, with

* Corresponding author.
*E-mail address:* zqin@hnu.edu.cn (Z. Qin).

https://doi.org/10.1016/j.future.2019.05.001
0167-739X/

the growing popularity of cloud computing, the secure search over encrypted cloud data becomes a research focus [11]. Some approaches have been proposed in [12–23], which aim to protect data and query privacies with better security, efficiency, and query experience. However, these works mainly consider the single data owner model, except [21]. In practice, a more normal scenario is that multiple data owners share their data on the cloud computing platform having the multi-tenant characteristic. A good example is the Personal Health Record sharing system like mymedwall.com, where multiple volunteer patients store their encrypted health data to the cloud server and only authorized users can perform searches over these data.

In a single-owner scheme, the sole data owner constructs secure searchable indexes for encrypted data files, which allows the cloud server to perform secure query according to query trapdoor (i.e., encrypted query keyword) submitted by an authorized data user. A data user obtains query trapdoor by asking the data owner for query keys or letting the online data owner generate query trapdoor for him.

Intuitively, the multi-owner scenario can be easily achieved by directly employing the single-owner model. For example, in the multi-owner model, all data owners share the same index encryption keys to build their secure searchable indexes. An authorized data user asks one of these data owners for query keys or trapdoors to request data files. Consequently, if one of the data owners carelessly leaks the index keys, the data security of all data owners would be compromised. Moreover, data owners may be unwilling to share secret keys each other in real life. Another solution seems to more reasonable, in which each data owner is allowed to use own secret keys to encrypt data indexes independently. In this case, a data user has to ask all data owners for query trapdoors or needs to maintain a large number of keys to generate legal query trapdoors. Obviously, this solution not only unavoidably brings heavy keys management [24], computation, and communication overhead, but also greatly reduces availability and flexibility of search system. Therefore, developing a full-fledged multi-owner scheme by directly employing the single-owner model will have many new challenges.

Though the research in [25] began to study the secure search problem based on the more challenging multi-owner model, the first real multi-owner secure search scheme was proposed in the recent work [8], which thoroughly discusses and analyzes the security challenges in the multi-owner model. However, to guarantee the security in the multi-owner model, the scheme needs to introduce an extra entity called *Administration Server* to re-encrypt the secure indexes. Moreover, the *Administration Server* has to store a pre-computed secret data on the cloud server and re-encrypt the user's query trapdoor every time to achieve effective search at the cloud side. As a result, once the *Administration Server* crashes, the whole search system will no more work. Therefore, the *Administration Server* not only increases the cost of the whole search system but also faces the problem of single point of failure or becomes a new attractive attack point for outer attackers.

### 1.2. Our contributions

In this paper, we propose a more practical multi-owner secure search scheme with multi-keyword conjunctive ranked query functionality. We make four key contributions which can be summarized as follows.

1. We propose an ingenious secure query scheme for multiple data owners scenario that allows each data owner to adopt a different temporary key to build secure index for each data file. An authorized data user does not need to know the temporary key and can instead use randomly chosen temporary keys to encrypt query keywords while the cloud can still correctly perform keyword matching on each encrypted index.

2. To speed up multi-keyword query and make the proposed scheme more efficient and practical, we design a query efficiency-improved multi-keyword conjunctive secure query scheme with the search complexity $\mathcal{O}(1)$ for each encrypted data file.

3. We design a query results ranking mechanism for multi-keyword top-$k$ query based on the encrypted keyword relevance score. For a multi-keyword query and a data file satisfying the query, the cloud can effectively compute the similarity between the data file and the query according to the individual encrypted relevance score of each query keyword to the data file.

4. We make the theoretical performance analysis and security proof and analysis for our proposed scheme. The extensive experiment evaluations demonstrate the effectiveness and efficiency of our proposed scheme.

The rest of this paper is organized as follows. We view the related work in Section 2. We state the problem and define notations in Section 3, an overview of our proposed scheme is given as well. In Section 4, we briefly introduce preliminaries used in our scheme. We present a basic secure search scheme for multiple data owners model in Section 5. Based on the basic secure search scheme, we construct a practical and efficient multi-keyword conjunctive query scheme in Section 6. For achieving ranking query, we propose a secure query results ranking mechanism in Section 7. Security proof and analysis are conducted in Section 8. We evaluate performances in Section 9 and conclude this paper in Section 10.

## 2. Related work

### 2.1. Conventional searchable encryption

Song et al. first implemented a practical technique in [4] that allows a server to perform secure search by linearly scanning the whole encrypted document using an encrypted query keyword. To improve search efficiency, in [5], Goh et al. made use of Bloom filter and pseudo-random functions to construct a secure searchable index for each encrypted data files and defined the search security model against adaptive chosen keyword attack. The security vulnerability of this scheme is to reveal the query privacy if keywords have been searched before. To further improve security and search efficiency, in [9], Curtmolaet al. adopted the inverted index and hash table to design a novel searchable encryption scheme and formally presented new and stronger security definitions. In [6], Boneh et al. first constructed a searchable encryption scheme under public-key setting. To improve user query experiences and enrich search functionalities, keyword conjunctive and disjunctive secure search were proposed in [7,8].

### 2.2. Secure search in cloud computing

Data outsourcing service promotes the further study on privacy-preserving search for cloud computing. Based on [9], Wang et al. [12] first used encrypted keyword relevance score as ranking criterion to implement single keyword top-$k$ secure search over encrypted cloud data. Cao et al. [14] used secure kNN computation scheme [26] based on the encrypted space vector model to construct a multi-keyword ranked search scheme, and in [16] Sun et al. further developed the scheme and made relevance ranking more accurate by introducing cosine measure based similarity scores. In [17], Sun et al. improved their work [16] and proposed a query results verifiable multi-keyword secure query scheme by constructing secure index tree and signature technique. Xia et al. [19] and Fu et al. [20] employ the

secure kNN technique to develop the dynamic multi-keywords ranked secure search and enabling personalized secure search over encrypted cloud data respectively to further improve the user search experience. To tolerate minor typos and enhance user search experience, fuzzy keyword search schemes over encrypted cloud data were proposed in [13,18,27,28]. Li et al. [29] proposed data deduplication based secure keyword search scheme in the cloud storage systems for reducing storage space and upload bandwidth. Other than the keyword-based search, Li et al. [30] considered the outsourced relational database and proposed a novel lightweight encryption mechanism supporting SQL-based queries. However, these schemes mainly consider single data owner model and directly extending them for multiple data owners model will bring heavy computation, communication, and key management overhead or incur many security problems described in Section 1. Though the research in [25] began to study the secure search problem based on the more challenging multi-owner model based on attribute-based encryption [31], the first formal multi-owner secure search scheme was proposed in the recent work [21], which comprehensively discusses and analyzes the security challenges in the multi-owner model. However, compared to general system model of secure query in cloud computing, the scheme needs to introduce an extra *Administration Server*. Our original work in [32] for multi-owner secure search scheme without the extra entity *Administration Server* is proposed to reduce the system complexity with the same security strength and better performance.

## 3. Problem formulation

In this section, we first present a system model and a threat model. Then, we elucidate security requirements of our scheme in detail. Finally, we define notations used to describe our scheme and give an overview of the multi-keyword ranked secure search for multiple data owners.

### 3.1. System model

We consider the multi-owner system architecture of search over encrypted cloud data, which includes multiple data owners, multiple data users, and the cloud server, as illustrated in Fig. 1. The cloud server provides storage and computation services for cloud customers in a pay-as-you-go fashion. In the system, each data owner encrypts data files and constructs encrypted searchable index for the security and searchability of data files simultaneously, and then outsources them to the cloud server. Only an authorized data user is allowed to search data files by submitting a query trapdoor and an optional number $k$ to the cloud server. Upon receiving the query trapdoor and $k$, the cloud server is responsible for performing search on encrypted indexes and rank query results according to certain ranked criteria. When the query ends, the top-$k$ most relevant encrypted data files are returned to the corresponding data user.

### 3.2. Threat model

The semi-trusted threat model was first proposed by Canetti et al. in [33] and has been widely adopted in the secure search field under the cloud computing environment, in which the cloud server is modeled an "honest-but-curious" semi-trusted threat entity. That is, the cloud server promises that it will always strictly obey data escrow protocols and correctly fulfill the functional responsibilities; however, we still cannot exclude the possibility that the cloud server wants to infer some useful information from outsourced data files and submitted query requests because of "curiosities" as the cloud server is a remote entity.

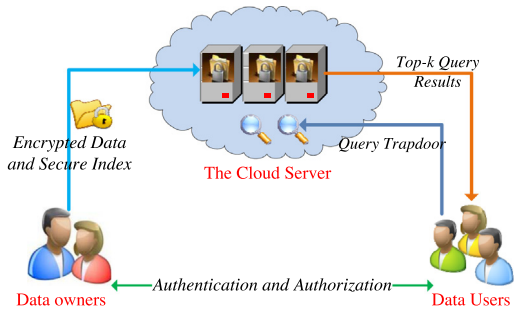



**Fig. 1.** A system model of search over encrypted data on a remote cloud center.

### 3.3. Security requirements

A secure search scheme should protect data owner's data security as well as the data user's query privacy from the semi-trusted cloud server. In the multiple data owners scenario, we propose that the secure search system should satisfy the following security requirements.

- **Security of Data Files**: Preventing the cloud server from obtaining data contents by accessing the outsourced data files themselves is the most primary security goal.
- **Security of Searchable Indexes**: A file index is created based on the keywords extracted from a data file, which directly reflects the contents of the data file. Therefore, the cloud server is prohibited from obtaining any useful information about data files by analyzing their indexes. Such useful information may include: how many identical keywords are contained in two different data files, whether some specific keywords are contained in certain data file, and the number of keywords of the data file. Moreover, in the multiple data owners environment, the leakage of index contents of one data owner should not compromise the security of other data owners. We propose that the most ideal encrypted index construction should be that even for totally same data files, their indexes are totally different yet with the same length while keeping uniform searchable ability. In this paper, we will construct such an index construction.
- **Trapdoor Privacy**: A trapdoor is the encrypted form of query keyword(s). The trapdoor privacy requires that the cloud server cannot recover the underlying plaintext keyword(s) according to the trapdoor.
- **Trapdoor Unlinkability**: For preventing the cloud server from knowing the relationship of trapdoors, the same query keywords should bear totally different trapdoors every time. The trapdoor unlinkability requires that the trapdoor generation algorithm should be randomized rather than deterministic.

### 3.4. Notions and overview

In this subsection, we define the notations used in this paper in Table 1 and give an overview of our scheme as shown in Fig. 2.

We give a concrete example to illustrate the main idea of our proposed scheme. For each data file $F_{i,j} \in \mathcal{F}_i$ owned by $\mathcal{DO}_i$, $\mathcal{DO}_i$ extracts keywords from $F_{i,j}$ to get its keyword set $W_{i,j}$ by case folding, stemming, and getting rid of stop words. To enable efficient and accurate ranked query, $\mathcal{DO}_i$ calculates the relevance score of each keyword to $F_{i,j}$ in $W_{i,j}$. $F_{i,j}$ is encrypted by a symmetric encryption such as AES under certain key. For $W_{i,j}$, $\mathcal{DO}_i$ uses a temporary secret key $sk_{i,j}$ to encrypt its keywords and relevance scores to generate the secure index $\mathcal{I}_{F_{i,j}}$ and then destroys the key

**Table 1**
Notations used in our scheme.

| Notation | Description |
|---|---|
| $\mathcal{DO}$; $\mathcal{DO}_i$ | A set of data owner, $\mathcal{DO} = \{\mathcal{DO}_1, \ldots \mathcal{DO}_{\lvert\mathcal{DO}\rvert}\}$ ; $i$th data owner in $\mathcal{DO}$, $\mathcal{DO}_i \in \mathcal{DO}$. |
| $\mathcal{F}_i$; $F_{i,j}$ | Data file set of $\mathcal{DO}_i$, $\mathcal{F}_i = \{F_{i,1}, \ldots, F_{i,\lvert\mathcal{F}_i\rvert}\}$; $j$th data file in $\mathcal{F}_i$, $F_{i,j} \in \mathcal{F}_i$. |
| $\mathcal{C}_i$; $C_{i,j}$ | Ciphertext set of $\mathcal{F}_i$, $\mathcal{C}_i = \{C_{i,1}, \ldots, C_{i,\lvert\mathcal{F}_i\rvert}\}$; ciphertext of $F_{i,j}$ , $C_{i,j} \in \mathcal{C}_i$. |
| $W_{i,j}$ | Set of distinct keyword of $F_{i,j}$, $W_{i,j} = \{w_{i,j,1}, \ldots, w_{i,j,\lvert W_{i,j}\rvert}\}$. |
| $\mathcal{I}_{(F_{i,j})}$ | Secure index of data file $F_{i,j}$. |
| $sk_{i,j}$ | Temporary index encryption key of $\mathcal{I}_{(F_{i,j})}$. |
| $\mathcal{Q}$ | A conjunctive query $\mathcal{Q} = \{w_1, \ldots, w_{\lvert\mathcal{Q}\rvert}\}$, $\lvert\mathcal{Q}\rvert > 1$. |
| $u$ | An authorized data user. |
| $r_u$, $q_u$ | Temporary trapdoor encryption keys of $u$. |
| $\mathcal{T}(w)_u$ | Trapdoor of query keyword $w$ submitted by $u$. |
| $\mathcal{T}(\mathcal{Q})_u$ | Trapdoor of $\mathcal{Q}$ submitted by $u$. |
| $\mathcal{RS}(w, F_{i,j})$ | Relevance score of a keyword $w \in W_{i,j}$ to its data file $F_{i,j}$. |
| $sim(\mathcal{Q}, F_{i,j})$ | Similarity score of the data file $F_{i,j}$ to $\mathcal{Q}$,where $F_{i,j}$ is a query result of $\mathcal{Q}$. |
| $\lvert\alpha\rvert$ | If $\alpha$ is a string, $\lvert\alpha\rvert$ denotes the bit length of $\alpha$; If $\alpha$ is a set, $\lvert\alpha\rvert$ denotes the cardinality of $\alpha$. |

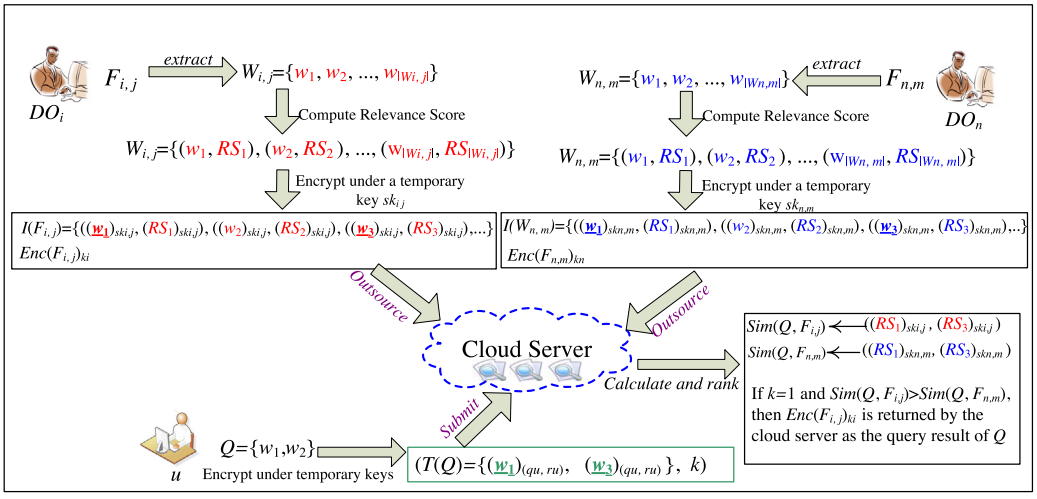



**Fig. 2.** An overview of the multi-keyword ranked secure search for multiple data owners.

$sk_{i,j}$ immediately. Given a $\mathcal{Q}$, an authorized data user $u$ chooses two temporary query keys $q_u$, $r_u$ to encrypted $\mathcal{Q}$ to get $\mathcal{T}(\mathcal{Q})_u$ and then destroys $q_u$ and $r_u$ immediately. Finally, $u$ submits $\mathcal{T}(\mathcal{Q})_u$ and a number $k$ to the cloud server. Upon receiving the $\mathcal{T}(\mathcal{Q})_u$, the cloud server searches over the index $\mathcal{I}_{F_{i,j}}$ and calculates the similarity score between $\mathcal{Q}$ and $F_{i,j}$ as $sim(\mathcal{Q}, F_{i,j})$ if $F_{i,j}$ is a query result of $\mathcal{Q}$. Ultimately, the cloud server sorts similarity scores of all query results of $\mathcal{Q}$ and returns top-$k$ encrypted data files to $u$.

## 4. Preliminaries

In this section, we briefly introduce several important techniques and security assumptions that will be used in this paper.

### 4.1. Bilinear pairing map

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ denote two cyclic multiplicative groups of order $q$. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfies the following properties:

- Computable: For any $Q, Z \in \mathbb{G}_1$, there is a polynomial time algorithm to compute $e(Q, Z) \in \mathbb{G}_2$.
- Bilinear: For all $x, y \in \mathbb{Z}_q^*$ and $Q, Z \in \mathbb{G}_1$, the equality $e(Q^x, Z^y) = e(Q, Z)^{xy}$ holds.
- Non-degenerate: If $g, h$ are generators of $\mathbb{G}_1$, then $e(g, h)$ is a generator of $\mathbb{G}_2$.

### 4.2. Discrete logarithm problem (DLP)

*DLP*: Let $g$ represent a generator of the group $\mathbb{G}$ with order $q$, given $g$ and $g^a$, compute $a \in \mathbb{Z}_q^*$. For any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, the advantage that $\mathcal{A}$ solves DLP is defined as:

$$Adv_{\mathcal{A}}^{\mathrm{DLP}} = Pr[\mathcal{A}(g, g^a) = a | a \in \mathbb{Z}_q^*]$$

*DLP Assumption*: For any PPT algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}^{\mathrm{DLP}}$ is negligible.

### 4.3. Decisional diffie-hellman problem (DDHP)

*DDHP*: Let $g$ represent a generator of the group $\mathbb{G}$ with order $q$. There are three random elements $a, b, c$ in $\mathbb{Z}_q^*$. Given $(g, g^a, g^b)$, distinguish the valid element $g^{ab}$ from the random element $g^c$ (i.e., decide whether $ab = c$). A PPT algorithm $\mathcal{A}$ has an advantage $Adv_{\mathcal{A}}^{\mathrm{DDHP}}$ in solving DDHP if:

$$Adv_{\mathcal{A}}^{\mathrm{DDHP}} \leq |Pr[\mathcal{A}(g^a, g^b, g^{ab}) = 1] - Pr[\mathcal{A}(g^a, g^b, g^c) = 1]|$$

*DDHP assumption*: for any PPT algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}^{\mathrm{DDHP}}$ is negligible.

## 5. Secure search over encrypted cloud data for multiple data owners

In this section, we construct a basic secure search scheme for multiple data owners model (SSMDO) that satisfies the proposed security requirements in the multi-owner scenario. Through the

theoretical performance analysis, we find the scheme is inefficient for enforcing multi-keyword conjunctive query and improve it in Section 6.

From a system-level perspective, SSMDO scheme mainly includes four phases: the system initialization phase, the secure *Index* construction phase, the query *Trapdoor* generation phase, and the secure *Search* phase. We give the detailed implementation of each phase in the following subsections.

### 5.1. System initialization phase

The system is fed a sufficiently large security parameter $l$ and then the phase sets up the work environment for our scheme. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic multiplicative groups with the same composite order $q$ and $g$ be a generator of $\mathbb{G}_1$. Define a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ and a cryptographic one-way hash function $H : \{0,1\}^* \to \mathbb{Z}_q^*$ which hashes an arbitrary string to an element in $\mathbb{Z}_q^*$. Meanwhile, the phase also generates two $l$-bit large primes $q_1, q_2 \in \mathbb{Z}_q$ that satisfy $q = q_1 \cdot q_2$. An efficient algorithm can be found in [34] to generate $e, \mathbb{G}_1, \mathbb{G}_2, H, q, q_1, q_2$ and the security strength of these parameters is determined implicitly by $l$.

All data owners secretly share parameters $q_1$ and $q_2$. We assume that the rigorous *data owner authorization* needs to be enforced before a new data user joins in the system. Once passing the *data owner authorization*, the authorized data user obtains the data file decryption keys and the important parameter $q_2$ from data owners via secure communication channels.

### 5.2. Constructing data secure index for each data owner

To make the outsourced encrypted data files searchable, each data owner $DO_i$ constructs a secure searchable index for each data file $F_{i,j} \in \mathcal{F}_i$ according to the corresponding keyword set $W_{i,j} = \{w_{i,j,1}, w_{i,j,2}, \ldots, w_{i,j,|W_{i,j}|}\}$. In the multiple data owners setting, compared with the single owner model, the most important challenge is that the different data owner should be able to choose different secret key to construct secure searchable index for system flexibility and data security. To satisfy this requirement, $DO_i$ constructs secure index for each data file $F_{i,j}$ as follows. Given the keyword set $W_{i,j}$, $DO_i$ first secretly chooses a temporary key $sk_{i,j}$ from $\mathbb{Z}_q^*$ at random and then encrypts each keyword $w_{i,j,h}(1 \le h \le |W_{i,j}|)$ as $g^{H(w_{i,j,h})+q_1 \cdot sk_{i,j}} \in \mathbb{G}_1$ to generate the secret index of $F_{i,j}$:

$$\mathcal{I}_{(F_{i,j})} = (g^{H(w_{i,j,1})+q_1 \cdot sk_{i,j}}, g^{H(w_{i,j,2})+q_1 \cdot sk_{i,j}}, \ldots, g^{H(w_{i,j,|W_{i,j}|})+q_1 \cdot sk_{i,j}})$$

After generating $\mathcal{I}_{(F_{i,j})}$, $DO_i$ destroys $sk_{i,j}$ immediately.

However, the index construction leaks the number of keywords contained in each data file. You can count the number of group elements in $\mathcal{I}_{(F_{i,j})}$ to obtain the information. To avoid revealing the number of keywords in each data file, like [9,12], random elements padding is necessary. Let $|W|_{max}$ be the maximum number of keywords contained in a data file. If $|W_{i,j}| < |W|_{max}$, $DO_i$ randomly chooses $|W|_{max} - |W_{i,j}|$ elements $R_1, \ldots, R_{|W|_{max}-|W_{i,j}|}$ from $\mathbb{G}_1$ and pads them to the $\mathcal{I}_{(F_{i,j})}$. At last, the constructed secure index of $F_{i,j}$ can be represented as:

$$\mathcal{I}_{(F_{i,j})} = (\underbrace{g^{H(w_{i,j,1})+q_1 \cdot sk_{i,j}}, \ldots, g^{H(w_{i,j,|W_{i,j}|})+q_1 \cdot sk_{i,j}}}_{F_{i,j}}, \underbrace{R_1, R_2, \ldots, R_{|W|_{max}-|W_{i,j}|}}_{padding})$$

### 5.3. Generating query trapdoor for data user

An authorized data user is allowed to encrypt query keywords of interest using randomly chosen secret key for every time request to achieve trapdoor privacy and trapdoor unlinkability. More specifically, given a keyword $w$, $u$ randomly chooses a temporary key $q_u$ from $\mathbb{Z}_q^*$ and encrypts $w$ as $g^{H(w) \cdot q_u}$. To effectively implement query, $u$ further computes $g^{q_2 \cdot \widehat{q_u}}$ and $g^{q_2}$, where $\widehat{q_u}$ is an inverse of $q_u$ such that $q_u \cdot \widehat{q_u} \mod q = 1$. The trapdoor of $w$ can be denoted as $\mathcal{T}_u(w) = (g^{H(w) \cdot q_u}, g^{q_2 \cdot \widehat{q_u}}, g^{q_2})$. However, the value $g^{q_2}$ always keeps unchanged every time. To eliminate the deterministic information, $u$ randomly chooses another temporary element $r_u$ from $\mathbb{Z}_q^*$ and generates the trapdoor of the keyword $w$ as follows:

$$\mathcal{T}_u(w) = (g^{H(w) \cdot q_u}, g^{q_2 \cdot r_u \cdot (\widehat{q_u}+1)}, g^{r_u \cdot q_2})$$

After generating the trapdoor $\mathcal{T}_u(w)$, $u$ destroys $q_u$ and $r_u$ immediately. For convenience of writing, we let $T_1 = g^{H(w) \cdot q_u}, T_2 = g^{q_2 \cdot r_u \cdot (\widehat{q_u}+1)}$ and $T_3 = g^{r_u \cdot q_2}$; thus the query trapdoor of keyword $w$ can be denoted as

$$\mathcal{T}_u(w) = (T_1, T_2, T_3)$$

Given a query keyword set $\mathcal{Q} = \{w_1, w_2, \ldots, w_{|\mathcal{Q}|}\}$, the trapdoor of $\mathcal{Q}$ can be denoted as follows:

$$\mathcal{T}_u(\mathcal{Q}) = \begin{cases} T_1 = \{g^{H(w_1) \cdot q_u}, g^{H(w_2) \cdot q_u}, \ldots, g^{H(w_{|\mathcal{Q}|}) \cdot q_u}\} \\ T_2 = g^{q_2 \cdot r_u \cdot (\widehat{q_u}+1)} \\ T_3 = g^{r_u \cdot q_2} \end{cases}$$

### 5.4. Search over all encrypted data files outsourced by different data owners for cloud server

The cloud server is equipped with powerful storage and computation capabilities, besides storing all data owners' encrypted data and secure indexes, it is also responsible for performing search on behalf of data users. More specifically, upon receiving the query trapdoor $\mathcal{T}_u(w) = \{T_1, T_2, T_3\}$, given the secure index $\mathcal{I}_{(F_{i,j})}$, the cloud server first computes

$$\begin{aligned} \mathcal{V}(\mathcal{T}_u(w)) &= e(T_1, T_2)/e(T_1, T_3) \\ &= e(g^{H(w) \cdot q_u}, g^{q_2 \cdot (\widehat{q_u}+r_u)})/e(g^{H(w) \cdot q_u}, g^{q_2 \cdot r_u}) \end{aligned}$$

If it can find an element $g^{H(w_{i,j,h})+q_1 \cdot sk_{i,j}}$ in $\mathcal{I}_{(F_{i,j})}$ that satisfies the following equality:

$$e(g^{H(w_{i,j,h})+q_1 \cdot sk_{i,j}}, T_3) = \mathcal{V}(\mathcal{T}_u(w))$$

then the data file $F_{i,j}$ is a correct search result. We say that if the query keyword $w$ satisfies $w = w_{i,j,h}, w_{i,j,h} \in F_{i,j}$, the above equality holds, the search correctness can be verified as follows.

$$\begin{aligned} e(g^{H(w_{i,j,h})+q_1 \cdot sk_{i,j}}, T_3) &= e(g^{H(w_{i,j,h})} \cdot g^{q_1 \cdot sk_{i,j}}, g^{r_u \cdot q_2}) \\ &= e(g^{H(w_{i,j,h})}, g^{r_u \cdot q_2}) \cdot e(g^{q_1 \cdot sk_{i,j}}, g^{r_u \cdot q_2}) \\ &= e(g^{H(w_{i,j,h})}, g^{r_u \cdot q_2}) \cdot 1 \\ &= e(g^{H(w_{i,j,h})}, T_3) \end{aligned}$$

$$\begin{aligned} \mathcal{V}(\mathcal{T}_u(w)) &= \frac{e(g^{H(w) \cdot q_u}, g^{q_2 \cdot r_u \cdot (\widehat{q_u}+1)})}{e(g^{H(w) \cdot q_u}, g^{q_2 \cdot r_u})} \\ &= \frac{e(g^{H(w) \cdot q_u}, g^{r_u \cdot q_2 \cdot \widehat{q_u}}) \cdot e(g^{H(w) \cdot q_u}, g^{q_2 \cdot r_u})}{e(g^{H(w) \cdot q_u}, g^{q_2 \cdot r_u})} \\ &= e(g^{H(w)}, g^{r_u \cdot q_2})^{q_u \cdot \widehat{q_u}} \\ &= e(g^{H(w)}, T_3) \end{aligned}$$

Because $e(g, g)$ is a group element in $\mathbb{G}_2$ with the order $q = q_1 q_2$, therefore $e(g,g)^{sk_{i,j} \cdot r_u \cdot q_1 \cdot q_2} = e(g,g)^{sk_{i,j} \cdot r_u \cdot q} = e(g,g)^{[sk_{i,j} \cdot r_u \cdot q \mod q]} = e(g,g)^0 = 1$. Obviously, $e(g^{H(w_{i,j,h})+q_1 \cdot sk_{i,j}}, T_3) = \mathcal{V}(\mathcal{T}_u(w))$ holds if $w = w_{i,j,h}$.

### 5.5. Performance analysis

To evaluate the computation cost and communication cost of the SSMDO, we first define some necessary notations. Let $P$ be a pairing operation and $H$ be a hash operation that hashes an arbitrary length string to an element in $\mathbb{Z}_q^*$. $E_{\mathbb{G}_1}$ denotes the exponentiation operation in $\mathbb{G}_1$, $M_{\mathbb{G}_1}$ and $M_{\mathbb{G}_2}$ denote the group multiplication in $\mathbb{G}_1$ and $\mathbb{G}_2$, $M_{\mathbb{Z}_q}$ and $A_{\mathbb{Z}_q}$ denote the multiplication and addition in $\mathbb{Z}_q$, respectively. Without loss of generality, we use $W_{i,j}$ to denote the $j$th data files of data owner $DO_i$ and use $|W|_{max}$ to denote the total number of elements contained in per-index after padding, the $\mathcal{Q}$ denotes a query keyword set. For ease of reading, we describe the computation cost, asymptotic complexity and output size of each phase in Table 2.

Note that Table 2 only displays the computation cost of one index construction, which is linear to the number of distinct keywords contained in the data file. Obviously, the total computation cost of secure indexes construction is $\sum_{i=1}^{|\mathcal{DO}|}\sum_{j=1}^{|\mathcal{F}_i|}(|W_{i,j}|(H + A_{\mathbb{Z}_q} + E_{\mathbb{G}_1}) + M_{\mathbb{Z}_q})$ and total output size of secure indexes is $\sum_{i=1}^{|\mathcal{DO}|}\sum_{j=1}^{|\mathcal{F}_i|}(|W|_{max}|\mathbb{G}_1|)$. The *search* performs keywords matching operations between trapdoors and secure indexes. If the linear search is employed, the average search times is $\frac{1+|W|_{max}}{2}$ for each query keyword in $\mathcal{Q}$ and $(2 + \frac{1+|W|_{max}}{2})|\mathcal{Q}|$ pairing operations are executed on one data index. Therefore, assume that, when the query $\mathcal{Q}$ ends, $n$ encrypted data files satisfy this query, the total search cost can be denoted as:

$$2P|\mathcal{Q}| + \sum_{i=1}^{|\mathcal{DO}|}\sum_{j=1}^{|\mathcal{F}_i|}\Big(\frac{1+|W|_{max}}{2}\Big)P|\mathcal{Q}|$$

## 6. Practical and efficient multi-keyword conjunctive query over encrypted data in the multi-owner model

In Section 5, we have implemented a secure keyword matching scheme for multi-owner model, SSMDO. However, the theoretical performance analysis tell us, given a multi-keyword conjunctive query $\mathcal{Q}$, that the cloud server needs $|\mathcal{Q}|$ rounds linear search and performs $(2 + \frac{1+|W|_{max}}{2})|\mathcal{Q}|$ pairing operations on one secure index for a successful query, which certainly will discount the query efficiency greatly due to the large number of pairing computations and makes the multi-keyword conjunctive secure query extremely inefficient. In other word, SSMDO is a single keyword secure query scheme in essence. Inspired by [7], in this section, we construct a practical multi-keyword conjunctive secure search scheme (MKSSMDO) based on SSMDO and significantly improve the query efficiency by largely reducing the pairing operations from $(2 + \frac{1+|W|_{max}}{2})|\mathcal{Q}|$ to 3. Correspondingly, the query complexity is significantly reduced from $\mathcal{O}((|W|_{max} + 1)|\mathcal{Q}|)$ to $\mathcal{O}(1)$.

### 6.1. MKSSMDO construction

To achieve the goal, we slightly change the secure index construction of SSMDO and improve the trapdoor generation and search algorithms. In MKSSMDO, there needs a pre-defined keyword dictionary. We use notation $\mathcal{D}$ to denote such a pre-defined keyword dictionary maintained by data owners and data users, in which the keywords are arranged in a random order. For ease of understanding, we give an example to illustrate how to construct the improved secure index. Given the keyword set $W_{i,j}$ of the data file $F_{i,j}$, the data owner $DO_i$ builds the secure index for $F_{i,j}$ as follows: for each keyword $w \in W_{i,j}$, $DO_i$ encrypts $w$ in the position that $w$ appears in $\mathcal{D}$; other positions are padded using random group elements. For example, assume $\mathcal{D} = \{w_1, w_2, w_3, w_4, w_5, w_6\}$ and the data file $F$'s keyword set is $W = \{w_2, w_4\}$, the secure index $\mathcal{I}_F$ can be denoted as:

$$\mathcal{I}_F = (R_1, g^{H(w_2)+q_1\cdot sk}, R_3, g^{H(w_4)+q_1\cdot sk}, R_5, R_6)$$

where $sk$ is a temporary key for $F$ and $R_1, R_3, R_5$, and $R_6$ are randomly chosen group elements in $\mathbb{G}_1$. Obviously, the secure indexes of all data files have the same size and consists of $|\mathcal{D}|$ random group elements in $\mathbb{G}_1$ each.

If a data user $u$ adopts the multi-keyword conjunctive query $\mathcal{Q} = \{w_1, \ldots, w_{|\mathcal{Q}|}\}$ to request encrypted data files, he generates the query trapdoor based on $\mathcal{D}$ as follows:

$$\mathcal{T}_u(\mathcal{Q}) = \begin{cases} T_1 = \{g^{q_u\cdot\sum_{k=1}^{|\mathcal{Q}|}(H(w_k))}, p_{w_1}, p_{w_2}, \ldots, p_{w_{|\mathcal{Q}|}}\} \\ T_2 = g^{q_2\cdot r_u\cdot(\widehat{q_u}+1)} \\ T_3 = g^{r_u\cdot q_2} \end{cases}$$

where $p_{w_k}$ points the position that $w_k$ appears in the universal set $\mathcal{D}$.

Upon receiving the $\mathcal{T}_u$, given the secure index $\mathcal{I}_{F_{i,j}}$, the cloud takes all group elements from positions $p_{w_1}, \ldots, p_{w_{|\mathcal{Q}|}}$ and checks whether the following equality holds or not:

$$e\Big(\prod_{k=1}^{|\mathcal{Q}|} g^{H(w_{i,j,p_{w_k}})+q_1\cdot sk_{i,j}}, T_3\Big) = e(T_1, T_2)/e(T_1, T_3)$$

If the equality holds, then the data file $F_{i,j}$ is a query result of $\mathcal{T}_u$. The query correctness can be easily verified as follows:

$$\begin{aligned} e(T_1, T_2)/e(T_1, T_3) &= \frac{e\Big(g^{q_u\cdot\sum_{k=1}^{|\mathcal{Q}|}(H(w_k))}, g^{q_2\cdot r_u\cdot(\widehat{q_u}+1)}\Big)}{e\Big(g^{q_u\cdot\sum_{k=1}^{|\mathcal{Q}|}(H(w_k))}, g^{r_u\cdot q_2}\Big)} \\ &= \frac{e\Big(g^{\sum_{k=1}^{|\mathcal{Q}|}(H(w_k))}, g^{q_2\cdot r_u}\Big)^{q_u\cdot\widehat{q_u}}\cdot e\Big(g^{q_u\cdot\sum_{k=1}^{|\mathcal{Q}|}(H(w_k))}, g^{r_u\cdot q_2}\Big)}{e\Big(g^{q_u\cdot\sum_{k=1}^{|\mathcal{Q}|}(H(w_k))}, g^{r_u\cdot q_2}\Big)} \\ &= e\Big(g^{\sum_{k=1}^{|\mathcal{Q}|}(H(w_k))}, T_3\Big) \end{aligned}$$

$$\begin{aligned} e\Big(\prod_{k=1}^{|\mathcal{Q}|} g^{H(w_{i,j,p_{w_k}})+q_1\cdot sk_{i,j}}, T_3\Big) &= e\Big(g^{\sum_{k=1}^{|\mathcal{Q}|}H(w_{i,j,p_{w_k}})}\cdot g^{\sum_{k=1}^{|\mathcal{Q}|}q_1\cdot sk_{i,j}}, g^{r_u\cdot q_2}\Big) \\ &= e\Big(g^{\sum_{k=1}^{|\mathcal{Q}|}H(w_{i,j,p_{w_k}})}, g^{r_u\cdot q_2}\Big)\cdot e\Big(g^{\sum_{k=1}^{|\mathcal{Q}|}q_1\cdot sk_{i,j}}, g^{r_u\cdot q_2}\Big) \\ &= e\Big(g^{\sum_{k=1}^{|\mathcal{Q}|}H(w_{i,j,p_{w_k}})}, g^{r_u\cdot q_2}\Big)\cdot e(g,g)^{|\mathcal{Q}|\cdot q_1\cdot sk_{i,j}\cdot r_u\cdot q_2} \\ &= e\Big(g^{\sum_{k=1}^{|\mathcal{Q}|}H(w_{i,j,p_{w_k}})}, T_3\Big) \end{aligned}$$

Obviously, if $w_{(i,j,p_{w_k})} = w_k$ for all $k = 1, 2, \ldots, |\mathcal{Q}|$, then the above equality holds.

### 6.2. Performance analysis

We can see that the secure index generation of MKSSMDO bears the same time complexity as the SSMDO, both of which increase with the size of the keyword set. MKSSMDO scheme greatly improves the search efficiency by largely reducing the time-consuming pair operations with the search complexity $\mathcal{O}(1)$. However, it may require more storage space to store secure indexes compared to SSMDO due to $|\mathcal{D}| > |W|_{max}$. For a multi-keyword query set $\mathcal{Q}$, the relatively time-consuming group exponentiation operations of trapdoor generation are reduced from $|\mathcal{Q}| + 2$ to 3 compared with SSMDO. Correspondingly, the output size of the trapdoor is also reduced from $(2+|\mathcal{Q}|)|\mathbb{G}_1|$ to $3|\mathbb{G}_1|$. we

**Table 2**
The cost, complexity, and output size of each phase in the scheme.

| Phase | Computation Cost | Complexity | Output Size |
|---|---|---|---|
| *Index* | $\|W_{i,j}\|(H + A_{\mathbb{Z}_q} + E_{\mathbb{G}_1}) + M_{\mathbb{Z}_q}$ | $\mathcal{O}(\|W\|_{i,j})$ | $(\|W\|_{max})\|\mathbb{G}_1\|$ |
| *Trapdoor* | $\|\mathcal{Q}\|(E_{\mathbb{G}_1} + M_{\mathbb{Z}_q} + H) + 2(E_{\mathbb{G}_1} + M_{\mathbb{Z}_q})$ | $\mathcal{O}(\|\mathcal{Q}\|)$ | $(\|\mathcal{Q}\|+2)\|\mathbb{G}_1\|$ |
| *Search* | $(2 + \frac{1+\|W\|_{max}}{2})\|\mathcal{Q}\|P + M_{\mathbb{G}_2}\|\mathcal{Q}\|$ | $\mathcal{O}((\|W\|_{max}+1) \times \|\mathcal{Q}\|)$ | – |

**Table 3**
The cost, complexity, and output size of each phase in the scheme.

| Phase | Computation Cost | Complexity | Size |
|---|---|---|---|
| *Index* | $\|W_{i,j}\|(H + A_{\mathbb{Z}_q} + E_{\mathbb{G}_1}) + M_{\mathbb{Z}_q}$ | $\mathcal{O}(\|W\|_{i,j})$ | $\|\mathcal{D}\|\|\mathbb{G}_1\|$ |
| *Trapdoor* | $\|\mathcal{Q}\|(A_{\mathbb{Z}_q} + H) + 3(E_{\mathbb{G}_1} + M_{\mathbb{Z}_q})$ | $\mathcal{O}(\|\mathcal{Q}\|)$ | $3\|\mathbb{G}_1\|$ |
| *Search* | $3P + M_{\mathbb{G}_2}$ | $\mathcal{O}(1)$ | – |

describe the computation cost, asymptotic complexity and output size of the improved scheme in Table 3.

Assume that, when the query $\mathcal{Q}$ ends, $n$ encrypted data files satisfy this query, the total search cost is

$$\left(2 + \sum_{i=1}^{|\mathcal{DO}|} \sum_{j=1}^{|\mathcal{F}_i|}\right) P.$$

## 7. Secure multi-keyword ranked search based on encrypted relevant scores of keywords

To achieve accurate multi-keyword ranked search over the encrypted data, given a data file $F_{i,j}$ and its keyword set $W_{i,j} = \{w_{i,j,1}, \ldots, w_{i,j,|W_{i,j}|}\}$, for each keyword $w_{i,j,h}$, the data owner $DO_i$ uses the TF $\times$ IDF rule to compute the relevant score of $w_{i,j,h}$ to $F_{i,j}$ as $\mathcal{RS}(w_{i,j,h}, F_{i,j})$ (simply denoted as $\mathcal{RS}_{w_{i,j,h}}$) and adds them into the secure index of $F_{i,j}$ that allows cloud to use this information to rank query results. Now, based on the pre-defined keyword universal set $\mathcal{D}$, the constructed secure index can be denoted as:

$$\mathcal{I}_{(F_{i,j})} = ((g^{H(w_{i,j,1})+q_1 \cdot sk_{i,j}}, \mathcal{RS}_{w_{i,j,1}}), (R_2, R_2')\ldots,$$
$$(g^{H(w_{i,j,|W_{i,j}|})+q_1 \cdot sk_{i,j}}, \mathcal{RS}_{w_{i,j,|W_{i,j}|}}), (R_f, R_f'), \ldots)$$

If a data file does not contain a certain keyword, the corresponding position of its secure index is padded using two random group elements.

We use the same TF $\times$ IDF formula as [21] to compute a keyword's relevance score with respect to a data file as follows:

$$\mathcal{RS}(w, F_d) = \frac{1}{|F_d|} \cdot (1 + \ln f_{d,w}) \cdot \ln\left(1 + \frac{N}{f_w}\right)$$

where $f_{d,w}$ denotes the TF of keyword $w$ in the file $F_d$, $|F_d|$ is the length of file $F_d$, $N$ denotes the total number of data files, and $f_w$ denotes the number of data files that contain keyword $w$.

For a multi-keyword conjunctive query $\mathcal{Q} = \{w_1, w_2, \ldots, w_{|\mathcal{Q}|}\}$, $|\mathcal{Q}| > 1$, to enable effective ranked query based on relevance scores of keywords, typically, the similarity score of a data file $F_d$ to the query $\mathcal{Q}$ can be defined to be the arithmetic mean of relevance scores of all query keywords to the data file $F_d$ if $F_d$ satisfies the query $\mathcal{Q}$ (i.e., $F_d$ contains all query keywords in $\mathcal{Q}$).

$$sim(\mathcal{Q}, F_d) = \frac{\mathcal{RS}(w_1, F_d) + \cdots + \mathcal{RS}(w_{|\mathcal{Q}|}, F_d)}{|\mathcal{Q}|} = \sum_{i=1}^{|\mathcal{Q}|} \mathcal{RS}(w_i, F_d)/|\mathcal{Q}|$$

For example, assuming that there are two data files $F_{i,a}$ and $F_{i,b}$ satisfying query $\mathcal{Q}$, the cloud takes $\mathcal{RS}(w_1, F_{i,a}), \ldots, \mathcal{RS}(w_{|\mathcal{Q}|}, F_{i,a})$ and $\mathcal{RS}(w_1, F_{i,b}), \ldots, \mathcal{RS}(w_{|\mathcal{Q}|}, F_{i,b})$ from corresponding index $\mathcal{I}_{(F_{i,a})}$ and $\mathcal{I}_{(F_{i,b})}$ respectively, and computes $sim(\mathcal{Q}, F_{i,a})$ and $sim(\mathcal{Q}, F_{i,b})$. If $sim(\mathcal{Q}, F_{i,a}) > sim(\mathcal{Q}, F_{i,b})$ then $F_{i,a}$ is more relevant to the query $\mathcal{Q}$ than $F_{i,b}$.

However, the relevant scores cannot be directly stored in the plaintext form since they reflect the occurrence frequency of keywords in data files. To prevent the cloud server from obtaining useful information from data indexes by analyzing the relevant scores, the relevant scores as sensitive information should also be encrypted, which disable effectively search results ranking. In [12], Wang et al. adopted order preserving symmetric encryption (OPSE) [35] to hide the relevance score information while keeping the comparison ability between two encrypted number. However, their scheme only supports single keyword ranked search. Moreover, OPSE is normally the deterministic encryption scheme that causes that the same relevance score of a certain keyword in the different data files has the same ciphertext, which compromises the security in the multi-owner model. In this paper, our scheme should achieve the following three goals. First, the relevant score of each distinct keyword within a data file should be encrypted. Second, the same relevant score of a certain keyword in different data files should have the different ciphertext. Third, our scheme should achieve multi-keyword conjunctive ranked search based on the encrypted relevance scores of keywords.

The first and second goals can be achieved easily by using the same method as keyword encryption. For example, for each keyword $w_{i,j,h}$ and its relevant score $\mathcal{RS}_{w_{i,j,h}}$ with respect to the data file $F_{i,j}$, $DO_i$ encrypts them under a random temporary secret key $sk_{i,j}$ and generates ciphertext of $w_{i,j,h}$ and $\mathcal{RS}_{w_{i,j,h}}$ to be $g^{H(w_{i,j,h})+q_1 \cdot sk_{i,j}}$ and $g^{\mathcal{RS}_{w_{i,j,h}}+q_1 \cdot sk_{i,j}}$. We get the ultimate version of the secure index construction as follows:

$$\mathcal{I}_{(F_{i,j})} = (g^{H(w_{i,j,1})+q_1 \cdot sk_{i,j}}, g^{\mathcal{RS}_{w_{i,j,1}}+q_1 \cdot sk_{i,j}}), (R_2, R_2')\ldots,$$
$$(g^{H(w_{i,j,|W_{i,j}|})+q_1 \cdot sk_{i,j}}, g^{\mathcal{RS}_{w_{i,j,|W_{i,j}|}}+q_1 \cdot sk_{i,j}}, (R_f, R_f'), \ldots)$$

which contains $2|\mathcal{D}|$ random elements in $\mathbb{G}_1$. Finally, we will implement multi-keyword ranked search to achieve the third goal. Obviously, given a query $\mathcal{Q}$, to perform ranked query, the cloud must calculate the similarity score of each query result with respect to $\mathcal{Q}$, i.e., the arithmetic mean of relevance scores of all query keywords in a query result. However, in an index, the relevance score of each keyword is encrypted individually. So, how to compute the similarity of a query result with respect to $\mathcal{Q}$ according to encrypted relevant scores is a key challenge.

We give an example to illustrate how to address this key challenge. Assume that data file $F_{i,j}$ is a query result of $\mathcal{Q} = w_1 \wedge w_2 \wedge \cdots \wedge w_{|\mathcal{Q}|}$, the cloud first takes the corresponding encrypted relevant scores $g^{\mathcal{RS}(w_1,F_{i,j})+q_1 \cdot sk_{i,j}}, \ldots, g^{\mathcal{RS}(w_{|\mathcal{Q}|},F_{i,j})+q_1 \cdot sk_{i,j}}$ from index $\mathcal{I}_{(F_{i,j})}$ and then computes $\prod_{n=1}^{|\mathcal{Q}|} g^{\mathcal{RS}(w_n,F_{i,j})+q_1 \cdot sk_{i,j}}$ and $e(T_3, g) = G_{T_3}$. Then, the similarity score $sim(\mathcal{Q}, F_{i,j})$ can be computed efficiently by the following formula while the cloud does not know the individual relevance score of each query keyword in $F_{i,j}$, i.e., $\mathcal{RS}(w_1, F_{i,j}), \ldots, \mathcal{RS}(w_{|\mathcal{Q}|}, F_{i,j})$.

$$sim(\mathcal{Q}, F_{i,j}) = \frac{\log_{G_{T_3}} e\left(\prod_{n=1}^{|\mathcal{Q}|} g^{\mathcal{RS}(w_n,F_{i,j})+q_1 \cdot sk_{i,j}}, T_3\right)}{|T_1| - 1}$$

Similarly, the cloud computes the above similarity score for each query result of $\mathcal{Q}$ and ranks query results by sorting all these relevance scores in descending order.

Actually, the above similarity $sim(\mathcal{Q}, F_{i,j})$ of data file $F_{i,j}$ to $\mathcal{Q}$ exactly satisfies:

$$sim(\mathcal{Q}, F_{i,j}) = \frac{\log_{G_{T_3}} e\left(\prod_{n=1}^{|\mathcal{Q}|} g^{\mathcal{RS}(w_n, F_{i,j}) + q_1 \cdot sk_{i,j}}, T_3\right)}{|T_1| - 1} = \sum_{n=1}^{|\mathcal{Q}|} \mathcal{RS}(w_n, F_{i,j})/|\mathcal{Q}|$$

i.e., the arithmetic mean of relevance scores of all query keywords. We verify the fact by the following description and derivation.

$$\begin{aligned} & e\left(\prod_{n=1}^{|\mathcal{Q}|} g^{\mathcal{RS}(w_n, F_{i,j}) + q_1 \cdot sk_{i,j}}, T_3\right) \\ &= e\left(g^{\sum_{n=1}^{|\mathcal{Q}|} \mathcal{RS}(w_n, F_{i,j})} \cdot g^{\sum_{n=1}^{|\mathcal{Q}|} q_1 \cdot sk_{i,j}}, g^{r_u \cdot q_2}\right) \\ &= e\left(g^{\sum_{n=1}^{|\mathcal{Q}|} \mathcal{RS}(w_n, F_{i,j})}, g^{r_u \cdot q_2}\right) \cdot e\left(g^{\sum_{n=1}^{|\mathcal{Q}|} q_1 \cdot sk_{i,j}}, g^{r_u \cdot q_2}\right) \\ &= e\left(g^{\sum_{n=1}^{|\mathcal{Q}|} \mathcal{RS}(w_n, F_{i,j})}, g^{r_u \cdot q_2}\right) \cdot e(g, g)^{|\mathcal{Q}| \cdot q_1 \cdot q_2 \cdot sk_{i,j} \cdot r_u} \\ &= e(g, g^{r_u \cdot q_2})^{\sum_{n=1}^{|\mathcal{Q}|} \mathcal{RS}(w_n, F_{i,j})} \\ &= G_{T_3}^{\sum_{n=1}^{|\mathcal{Q}|} \mathcal{RS}(w_n, F_{i,j})} \end{aligned}$$

Because $\sum_{n=1}^{|\mathcal{Q}|} \mathcal{RS}(w_n, F_{i,j})$ is a very small number, the cloud can very efficiently compute by [36]:

$$\sum_{n=1}^{|\mathcal{Q}|} \mathcal{RS}(w_n, F_{i,j}) = \log_{G_{T_3}} e\left(\prod_{n=1}^{|\mathcal{Q}|} g^{\mathcal{RS}(w_n, F_{i,j}) + q_1 \cdot sk_{i,j}}, T_3\right)$$

Therefore,

$$\begin{aligned} & \log_{G_{T_3}} e\left(\prod_{n=1}^{|\mathcal{Q}|} g^{\mathcal{RS}(w_n, F_{i,j}) + q_1 \cdot sk_{i,j}}, T_3\right) / (|T_1| - 1) \\ &= \sum_{n=1}^{|\mathcal{Q}|} \mathcal{RS}(w_n, F_{i,j})/|\mathcal{Q}| \\ &= sim(\mathcal{Q}, F_{i,j}) \end{aligned}$$

The computation cost of computing similarity between the query $\mathcal{Q}$ and a query result is $2P + (|\mathcal{Q}| - 1)M_{\mathbb{G}_1} + T_{log} + T_{div}$, where $T_{log}$ denotes the time cost of a logarithm computation and $T_{div}$ denotes the time cost of an integer division operation.

## 8. Security proof and analysis

Data index security is the most important security goal for the cloud secure search scheme, our constructed secure index consists of encrypted keywords, encrypted relevances score, and random group elements. In this section, we first prove the security of the keyword and relevance score encryption and then provide a multianalysis according to the security requirements proposed in Section 3.

### 8.1. Security proof

In our secure index construction, both the keywords and relevance scores are converted into group elements by encrypting and the basic encryption block can be denoted as: $E(m) = g^{m+q_1 \cdot sk}$, where $sk$ is a randomly chosen secret value for encrypting $m$. We claim that the encryption scheme $E$ achieves semantical security against the chosen plaintext attack (CPA) under the DDHP assumption. Before proving the assertion, we first define an advantage that a polynomial adversary $\mathcal{A}$ is able to break $E$ by briefly playing a conventional challenger and adversary game as follows.

**Setup** The challenger runs the Setting Initialization algorithm and publishes public parameters to the adversary.

**Phase 1** The adversary is allowed to access encryption oracle $E$ for many times and inputs two messages $m_0$ and $m_1$,

**Challenge** The adversary sends $m_0$ and $m_1$ to the challenger. The challenger flips a random binary coin $b$ and encrypts $m_b$ as $E(m_b)$. The ciphertext $E(m_b)$ is sent to the adversary.

**Phase 2** The adversary continues to access the encryption oracle $E$.

**Guess** The adversary inputs the guess $b'$ of $b$.

$\mathcal{A}$'s advantage of winning the game is defined to be

$$Adv_{\mathcal{A}}^{\text{CPA}} = \left| Pr[b = b'] - \frac{1}{2} \right|$$

For any polynomial time adversary $\mathcal{A}$, if the advantage $Adv$ that $\mathcal{A}$ breaks an encryption is negligible, then the encryption scheme is semantically secure encryption against CPA .

**Theorem 1.** *Our proposed keyword and relevance score encryption $E$ is semantically secure if DDHP assumption holds.*

**Proof.** Suppose a polynomial time adversary $\mathcal{A}$ has a non-negligible advantage $\epsilon$ to win the above game to break the encryption $E$, we can construct an algorithm $\mathcal{B}$ who can solve the DDH problem with a non-negligible advantage.

The challenger $\mathcal{C}$ first flips a binary coin $\mu$. If $\mu = 0$, $\mathcal{C}$ sets tuple $t_0 : (g, A = g^a, B = g^b, C = g^{ab})$; if $\mu = 1$, he sets tuple $t_1 : (g, A = g^a, B = g^b, C = g^c)$, where $a, b, c$, are chosen from $\mathbb{Z}_q^*$ at random uniformly. Tuple $t_\mu$ is sent to simulator $\mathcal{B}$. The simulator $\mathcal{B}$ plays the following game with adversary $\mathcal{A}$ on behalf of challenger $\mathcal{C}$.

**Setup** $\mathcal{B}$ sends the public parameter $\{q, g\}$ to $\mathcal{A}$.

**Phase 1** $\mathcal{A}$ accesses the encryption function $E$ many times by using an arbitrary message in $\mathbb{Z}_q$ to ask the corresponding ciphertext every time. Finally, he outputs two messages $m_0$ and $m_1$ and sends them to $\mathcal{B}$.

**Challenge** $\mathcal{B}$ flips a binary coin $\gamma$ and encrypts message $m_\gamma$ as $\mathcal{E} = g^{m_\gamma} \cdot C$.

If $\mu = 0$, $C = g^{ab}$. As $sk$ is a randomly chosen element in encryption $E$, $sk \cdot q_1$ is also a random element, we let $ab = sk \cdot q_1$. Thus, we have $\mathcal{E} = g^{m_\gamma} \cdot C = g^{m_\gamma} \cdot g^{ab} = g^{m_\gamma + sk \cdot q_1}$. Therefore, $\mathcal{E}$ is a valid message encryption of $E$. If $\mu = 1$, $C = g^c$. Then we have $\mathcal{E} = g^{m_\gamma + c}$. Since $c$ is a random element, therefore $\mathcal{E}$ is a random element in $\mathbb{G}_1$ from $\mathcal{A}$'s perspective and contains no information about $m_\gamma$.

**Phase 2** $\mathcal{A}$ continues to ask the encryption oracle $E$.

**Guess** $\mathcal{A}$ outputs a guess $\gamma'$ of $\gamma$. If $\gamma' = \gamma$, then $\mathcal{B}$ outputs the guess $\mu' = 0$ of $\mu$. This means $\mathcal{C}$ sent the valid encryption tuple $t_0 : (g, A = g^a, B = g^b, C = g^{ab})$ to $\mathcal{B}$. Since $\mathcal{A}$ has advantage $\epsilon$ to break $E$, therefore, the probability $\mathcal{A}$ outputs guess $\gamma'$ of $\gamma$ satisfying $\gamma' = \gamma$ is $\frac{1}{2} + \epsilon$. Correspondingly, the probability that $\mathcal{B}$ outputs guess $\mu'$ of $\mu$ satisfying $\mu' = \mu = 0$ is $\frac{1}{2} + \epsilon$. If $\gamma' \neq \gamma$, then $\mathcal{B}$ outputs the guess $\mu' = 1$ of $\mu$. This means random tuple $t_1$ was sent to $\mathcal{B}$. Therefore, the probability $\mathcal{A}$ outputs guess $\gamma'$ of $\gamma$ satisfying $\gamma' = \gamma$ is $\frac{1}{2}$. Correspondingly, the probability that $\mathcal{B}$ outputs guess $\mu'$ of $\mu$ satisfying $\mu' = \mu = 1$ is $\frac{1}{2}$.

Hence, the overall advantage that $\mathcal{B}$ solves the DDHP can be computed:

$$\begin{aligned} adv_{\mathcal{B}}^{\text{DDHP}} &= \left| \frac{1}{2} Pr[\mu = \mu' | \mu = 0] + \frac{1}{2} Pr[\mu = \mu' | \mu = 1] - \frac{1}{2} \right| \\ &= \left| \left[ \frac{1}{2} \left( \frac{1}{2} + \epsilon \right) + \frac{1}{2} \cdot \frac{1}{2} \right] - \frac{1}{2} \right| = \frac{\epsilon}{2} \end{aligned}$$

Therefore, if $\mathcal{A}$ can break $E$ with a non-negligible advantage $\epsilon$, the $\mathcal{B}$ is able to solve the DDHP with the non-negligible advantage $\frac{\epsilon}{2}$, which contradicts the DDHP assumption. □

### 8.2. Security analysis

In this section, we provide an overall security analysis for our multi-owner secure query scheme.

- **Security of Data files**: The security of data files can be well protected by using the semantically secure symmetric encryption such as AES. As long as symmetric keys are kept secret from the cloud server, the cloud server cannot obtain the outsourced data.
- **Security of secure index**: In Theorem 1, we have proved that the keyword and relevance score encryption is a semantically secure encryption based on the DDHP assumption. On the other hand, our constructed indexes achieve the security requirements of multiple data owners scenario. More concretely, first, the same keywords in different data files have completely different ciphertexts. For example, given two data files $F_{i,a}$ and $F_{i,b}/F_{j,b}$ ($F_{i,*}$, $F_{j,*}$ are owned by $DO_i$ and $DO_j$, respectively), the cloud cannot observe how many the same keywords exist in $W_{i,a}$ and $W_{i,b}/W_{j,b}$ from their encrypted indexes $\mathcal{I}_{(F_{i,a})}$, $\mathcal{I}_{(F_{i,b})}/\mathcal{I}_{(F_{j,b})}$ due to different temporary keys $sk_{i,a}$ and $sk_{i,b}/sk_{j,b}$ such that they cannot be distinguished even if $W_{i,a} = W_{i,b}/W_{j,b}$. Second, for a data owner $DO_i$, the leakage of a temporary key does not endanger other data files. Third, a data owner leaks his temporary keys that would not lead to the data files disclosure of other owners. Lastly, the number of keywords contained in each data file is hidden by padding random elements. Therefore, the cloud server cannot obtain any useful information about the outsourced data by analyzing their secure indexes as long as the DDHP assumption holds.
- **Trapdoor Privacy and Trapdoor Unlinkability**: Given the trapdoor $\mathcal{T}_u(w)$ of keyword $w$, the cloud cannot recover $w$ from $T_1 = g^{H(w)\cdot q_u}$ due to requiring to solve the discrete logarithm problem in $\mathbb{Z}_q$ (DLP assumption). Moreover, large number factorization problem and the cryptography hash function $H$ further guarantee the security of the trapdoor. In addition, the random temporary keys $q_u, r_u$ ensure the security property of trapdoor unlinkability. Therefore, the cloud server cannot obtain any useful contents about search queries as long as the DLP is hard.
- **Query Security** In the process of query, the cloud server may try to obtain the underlying query keyword by computing $e(T_1, T_2)/e(T_1, T_3) = e(g, g)^{H(w)\cdot q_2 \cdot r_u}$. However, the cloud server cannot recover $w$ from $e(g, g)^{H(w)\cdot q_2 \cdot r_u}$ due to the DLP assumption, the cryptography hash function $H$, and secret value $q_2, r_u$. In addition, an unauthorized user cannot generate complete query trapdoor due to the lack of the key $q_2$ and thus cannot obtain effective query without $T_2$ and $T_3$.

## 9. Experimental evaluation

In this section, we experimentally evaluate the performance of SSMDO and MKSSMDO. To provide an effective and reasonable performance comparison, we also implement the off-the-peg multi-owner scheme, Privacy preserving Ranked Multi-keyword Search in a Multi-owner Model (PRMSM) [21]. Experiment results demonstrate that MKSSMDO has better and more practical search efficiency than SSMDO and PRMSM.

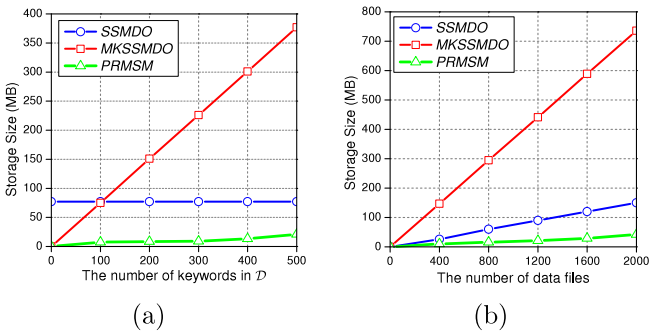



**Fig. 3.** (a) The storage overhead of secure indexes for different size of $\mathcal{D}$ with the same $|W|_{max} = 102$ and the same number of data files $n = 1000$. (b) The storage overhead of secure indexes for different number of data files with the same $|W|_{max} = 102$ and $|\mathcal{D}| = 500$.

### 9.1. Evaluation setup

We evaluate the performance of our secure scheme on the real Enron Email Dataset [37] and randomly select 2000 files to build our experimental subset, from which 738 keywords are extracted by using Hermetic Word Frequency Counter [38]. We generate keyword set for each data file, the minimal and maximal number of keywords in a certain data file is 47 and 102, respectively. Thus, we set the maximal number of keywords $|W|_{max} = 102$.

All experiment programs are developed on Java platform based on the JPBC library [39]. The *Type A1* elliptic curve is used in our experiments.

The software and hardware configurations are as follows. The client side is a Windows 7 desktop system with 2.3-GHz Intel Core (TM) i5-6200U and 4GB RAM , which is responsible for performing secure index and query trapdoor generation. The server side is also Windows 7 desktop system with 3.60-GHz Intel Core (TM) i7-7700 CPU and 8 GB RAM, which represents the cloud server to perform search over encrypted data.

### 9.2. Storage overhead of secure index

The secure index of each data file consists of $2|W|_{max}$ and $2|\mathcal{D}|$ group elements in SSMDO and MKSSMDO, respectively. PRMSM is an inverted index construction, which contains $2|\mathcal{D}|$ group elements and a set of data file identifier nodes. We invoke the API *getLengthInBytes()* to obtain the size, 386 bytes, of a group element in *Type A1* elliptic curve. Fig. 3(a) demonstrates that the secure index of SSMDO is independent of the size of $\mathcal{D}$ and consists of $2|W|_{max} = 204$ group elements, while each secure index of MKSSMDO and the inverted indexes of PRMSM consists of $2|\mathcal{D}|$ group elements and thus the storage size of their secure indexes is linear to the size of $\mathcal{D}$. When setting $|\mathcal{D}| = 500$, $|W|_{max} = 102$, and $n = 1000$, the index storage size of these schemes is 77 MB, 377 MB, and 21 MB, respectively. Fig. 3(b) demonstrates that, given $|W|_{max} = 102$ and $|\mathcal{D}| = 500$, the storage overhead of SSMDO, MKSSMDO, and PRMSM all increases linearly with an increasing number of data files. We can observe that MKSSMDO requires much more storage space than SSMDO and PRMSM to store secure indexes of data files. About 150 MB, 736 MB, and 42 MB storage spaces are needed when the number of data files achieves 2000 for SSMDO, MKSSMDO, and PRMSM respectively. They are acceptable for the cheap cloud storage nowadays. Note that PRMSM only needs 77 KB to save group elements when $|\mathcal{D}| = 500$, other storage spaces are used to store file identifier nodes.

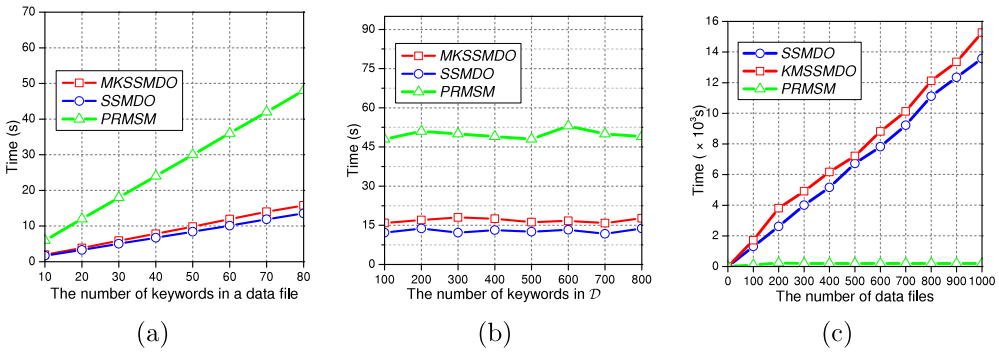



**Fig. 4.** (a) The time cost of secure index generation for different number of keywords contained in a data file with the fixed size of keyword dictionary $|\mathcal{D}| = 500$. (b) The time cost of secure index generation for different size of $\mathcal{D}$ with fixed the number of keywords contained in a data file $|W| = 80$. (c) The time cost of secure index generation for different number of data files with the fixed size of $|\mathcal{D}| = 500$ and fixed the number of keywords contained in a data file $|W| = 80$.

### 9.3. Time cost of secure index construction

Fig. 4(a) shows that the time cost of secure index construction for SSMDO, MKSSMDO, and PRMSM linearly increases when varying the number of keywords in a data file from 10 to 80 and MKSSMDO needs a little more time than SSMDO due to more random elements padding. We can observe that when the number of keywords achieves 80, the index construction time of SSMDO, MKSSMDO, and PRMSM is 13.5 s, 15.8 s, and 47.8 s, respectively. The reason, that the time cost of PRMSM is more than our scheme when constructing the secure index for one data file, is to require to generate a posting list for each keyword associated with the data file, which needs more exponentiation operations. Fig. 4(b) further illustrates that the time cost of secure index construction of these schemes is affected by the number of keywords of the data file only and is independent of the size of the keyword dictionary $\mathcal{D}$. Since SSMDO and MKSSMDO is a per-file based index, the time cost of constructing secure indexes is linear to the number of data files while PRMSM is insensitive to the size of data files for index construction due to adopting inverted index, as shown in Fig. 4(c). Moreover, the more the number of data files is, the more obvious the advantage of PRMSM is for index construction when fixing the size of the keyword dictionary. Though the index construction is a relatively expensive computation process in our scheme, it is a one-time operation for each data owner in practice.

### 9.4. Time cost of trapdoor generation

Fig. 5(a) shows the time cost of trapdoor generation for different number of query keywords when fixing the size of $\mathcal{D}$ to be 500. We can observe that the time cost of SSMDO and PRMSM grows linearly with the increasing number of query keywords while the time cost of MKSSMDO is not affected by the number of query keywords. This is because that, given a multi-keyword query $\mathcal{Q}$, SSMDO and PRMSM needs $2+|\mathcal{Q}|$ and $1+|\mathcal{Q}|$ exponentiation computations respectively, while MKSSMDO only needs 3 exponentiation computations for any number of query keywords. Fig. 5(b) shows the time cost of trapdoor generation for different size of $\mathcal{D}$ with the fixed number of query keywords. We can observe that the trapdoor generation time of the three schemes is independent of the size of the keyword dictionary $\mathcal{D}$.

It is worth emphasizing that we only test the time cost of trapdoor generation at the data user side in the above experiments. Compared with our schemes, PRMSM actually needs more time to generate complete query trapdoor since it further requires the *Administration Server* to re-encrypt the submitted trapdoor previously generated by a data user.

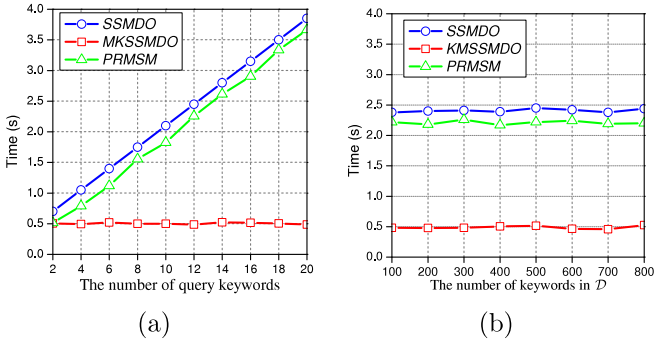



**Fig. 5.** (a) The time cost of trapdoor generation for different number of query keywords with the fixed size of keyword dictionary $|\mathcal{D}| = 500$. (b) The time cost of trapdoor generation for different size of keyword dictionary $\mathcal{D}$ with the fixed number of query keywords $|\mathcal{Q}| = 12$.

### 9.5. Time cost of secure search

We execute the secure search at the server side with more powerful computation capacities. As we can see from Fig. 6(a), the time cost of SSMDO and PRMSM is nearly linear to the number of query keywords while the time cost of MKSSMDO is not affected by the number of query keywords when fixing the size of data file set. Moreover, SSMDO needs to consume more time on keyword matching than PRMSM when fixing the size of keyword dictionary and data file set to be 200 and 1000, respectively. This is because that SSMDO and PRMSM needs to perform $1000|\mathcal{Q}|+2$ and $200|\mathcal{Q}|+1$ pair operations for a multi-keyword conjunctive query. Fig. 6(b) shows that the time cost of search for PRMSM increases linearly with the size of keyword dictionary $\mathcal{D}$ while SSMDO and MKSSMDO are insensitive to the size of keyword dictionary $\mathcal{D}$. Fig. 6(c) shows that, given the fixed size of query keywords and the keyword dictionary, the time cost of search for SSMDO and MKSSMDO grows linearly with the increasing number of data files while PRMSM generally remains unchanged. More importantly, we can observe from Fig. 6 that MKSSMDO needs to consume far less time on search than SSMDO and PRMSM, which further illustrates that MKSSMDO is a real and practical multi-keyword conjunctive query scheme. In addition, for MKSSMDO, the time cost of search can be further reduced by outsourcing search operations to the computationally powerful cloud server in practice.

## 10. Conclusion

In this paper, we investigate multi-keyword conjunctive ranked secure query technique over encrypted cloud data for

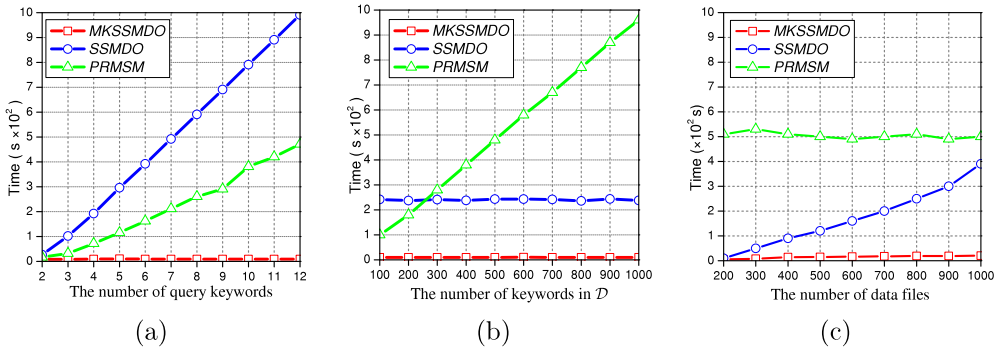



**Fig. 6.** (a) Time cost of search for different number of query keywords with the same size of keyword dictionary, $|\mathcal{D}| = 200$ and the same size of data file set, $n = 1000$. (b) Time cost of search for different size of keyword dictionary $\mathcal{D}$ with the same size of query keywords, $|\mathcal{Q}| = 5$ and the same size of data file set, $n = 1000$. (c) Time cost of search for different size of data files with the same number of query keywords, $|\mathcal{Q}| = 5$ and the same size of keyword dictionary, $|\mathcal{D}| = 500$.

multiple data owners scenario. We first construct a secure single keyword search scheme SSMDO for multi-owner model. To achieve a practical efficiency for multi-keyword conjunctive query, we develop SSMDO and propose a real multi-keyword conjunctive query scheme MKSSMDO with practical search efficiency. On the other hand, to achieve accurate query results ranking, our scheme allows the cloud to calculate the similarity scores of a multi-keyword conjunctive query to its query results based on individual encrypted keyword relevance score without knowing actual value of individual relevance score; based on similarity scores, the cloud can accurately and effectively rank query results and implement top-$k$ query. We theoretically analyze the performance and security of our proposed scheme and further experimentally evaluate the correctness and effectiveness in a real data set. A fuzzy multi-keyword conjunctive ranked query scheme for multi-owner model in the cloud computing will be explored in our future work.

## Acknowledgments

The authors would like to express their gratitude to the anonymous reviewers whose constructive comments have helped to improve the quality of the manuscript. The work is supported by the National Natural Science Foundation of China under Grant Nos. 61772191, 61772088, and 61472131; The Natural Science Foundation of Hunan Province, China under Grant No. 2017JJ2292; Science and Technology Key Projects of Hunan Province, China under Grant Nos. 2015TP1004 and 2016JC2012; Outstanding Youth Research Project of Provincial Education Department of Hunan, China under Grant No. 17B030; Open Research Fund of Hunan Provincial Key Laboratory of Network Investigational Technology, China under Grant No. 2016WLZC011; Science and Technology Planning Project of Changsha, China under Grant No. k1705018.

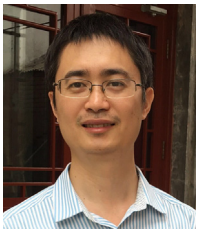
## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

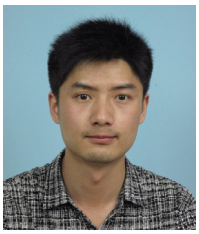
[1] T. Mather, S. Kumaraswamy, S. Latif, Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance, OReilly Media, 2009.
[2] S. Kamara, K. Lauter, Cryptographic cloud storage, in: Springer FC, 2010, pp. 136–149.
[3] K. Ren, C. Wang, Q. Wang, Security challenges for the public cloud, IEEE Internet Comput. 16 (1) (2012) 69–73.
[4] D. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: IEEE Symposium on Security and Privacy, Vol. 8, 2000, pp. 44–55.
[5] E.-J. Goh, Secure Indexes, Tech. Rep., IACR ePrint Cryptography Archive, 2003, http://eprint.iacr.org/2003/216.
[6] D. Boneh, G.D. Crescenzo, R. Ostrovsky, G. Persiano, Public-key encryption with keyword search, in: EUROCRYPR, 2004, pp. 506–522.
[7] P. Golle, J. Staddon, B. Waters, Secure conjunctive keyword search over encrypted data, in: Springer ACNS, 2004, pp. 31–45.
[8] L. Ballard, S. Kamara, F. Monrose, Achieving efficient conjunctive keyword searches over encrypted data, in: IEEE ICICS, 2005, pp. 414–426.
[9] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved deinitions and efficient constructions, in: ACM CCS, Vol. 19, 2006, pp. 79–88.
[10] M. Bellare, A. Boldyreva, A. O'Neill, Deterministic and efficiently searchable encryption, in: Springer CRYPTO, 2007, pp. 535–552.
[11] F. Han, J. Qin, J. Hu, Secure searches in the cloud: A survey, Future Gener. Comput. Syst. 62 (2016) 66–75.
[12] C. Wang, N. Cao, J. Li, K. Ren, W. Lou, Secure ranked keyword search over encrypted cloud data, in: IEEE ICDCS, 2010, pp. 253–262.
[13] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, W. Lou, Fuzzy keyword search over encrypted data in cloud computing, in: IEEE INFOCOM Mini-Conference, 2010, pp. 1–5.
[14] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, in: IEEE INFOCOM, 2011, pp. 829–837.
[15] M. Li, S. Yu, N. Cao, W. Lou, Authorized private keyword search over encrypted data in cloud computing, in: IEEE ICDCS, 2011, pp. 383–392.
[16] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, H. Li, Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, in: ACM ASIACCS, 2013.
[17] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 3025–3035.
[18] B. Wang, S. Yu, W. Lou, Y.T. Hou, Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud, in: IEEE INFOCOM, 2014, pp. 2112–2120.
[19] Z. Xia, X. Wang, X. Sun, Q. Wang, A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst. 27 (2) (2015) 340–352.
[20] Z. Fu, K. Ren, J. Shu, X. Sun, F. Huang, Enabling personalized search over encrypted outsourced data with efficiency improvement, IEEE Trans. Parallel Distrib. Syst. 27 (9) (2016) 2546–2559.
[21] W. Zhang, Y. Lin, S. Xiao, J. Wu, S. Zhou, Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing, IEEE Trans. Comput. 65 (5) (2016) 1566–1577.
[22] H. Yin, Z. Qin, L. Ou, K. Li, A query privacy-enhanced and secure search scheme over encrypted data in cloud computing, J. Comput. System Sci. 90 (2017) 14–27.
[23] R. Li, Z. Xu, W. Kang, K.C. Yow, C.Z. Xu, Efficient multi-keyword ranked query over encrypted data in cloud computing, Future Gener. Comput. Syst. 30 (1) (2014) 179–190.
[24] J. Li, X. Chen, M. Li, P. Lee, W. Luo, Secure deduplication with efficient and reliable convergent key management, IEEE Trans. Parallel Distrib. Syst. 25 (6) (2014) 1615–1625.

[25] W. Sun, S. Yu, W. Lou, Y.T. Hou, H. Li, Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud, in: IEEE INFOCOM, 2014, pp. 226–234.
[26] W. Wong, D. Cheung, B. Kao, N. Mamoulis, Secure knn computation on encrypted databases, in: ACM SIGMOD, 2009, pp. 139–152.
[27] P. Xu, H. Jin, Q. Wu, W. Wang, Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack, IEEE Trans. Comput. 62 (11) (2013) 2266–2277.
[28] F. Xhafa, J. Wang, X. Chen, J.K. Liu, J. Li, P. Krause, An efficient PHR service system supporting fuzzy keyword search and fine-grained access control, Soft Comput. 18 (9) (2014) 1795–1802.
[29] J. Li, X. Chen, F. Xhafa, L. Barolli, Secure deduplication storage systems supporting keyword search, J. Comput. System Sci. 81 (8) (2015) 1532–1541.
[30] J. Li, Z. Chen, X. Chen, F. Xhafa, D.S. Wong, L-EncDB: A lightweight framework for privacy-preserving data queries in cloud computing, Knowledge-based Systems 79 (2015) 18–26.
[31] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encryption data, in: ACM CCS, 2006, pp. 89–98.
[32] H. Yin, Z. Qin, J. Zhang, W. Li, L. Ou, Y. Hu, K. Li, Secure conjunctive multi-keyword search for multiple data owners in cloud computing, in: IEEE ICPADS, 2016, pp. 761–768.
[33] R. Canetti, U. Feige, O. Goldreich, M. Naor, Adaptively secure multi-part computation, in: ACM STOC, 1996, pp. 639–648.
[34] D. Boneh, E. Goh, K. Nissim, Evaluating 2-DNF formulas on ciphertexts, in: Springer TCC, 2005, pp. 325–342.
[35] A. Boldyreva, N. Chenette, Y. Lee, A. ONeill, Order-preserving symmetric encryption, in: Spinger EUROCRYPT, Vol. 5479, 2009, pp. 224–241.
[36] E. Teske, Computing discrete logarithms in arithmetic progressions, http://citeseerx.ist.psu.edu/.
[37] Enron dataset, http://nlp.cs.aueb.gr/software.
[38] H. Systems, Hermetic word frequency counter, http://www.hermetic.ch/wfc/wfc.htm.
[39] JPBC, http://gas.dia.unisa.it/projects/jpbc/index.html.



**Hui Yin** received his B.S. in Computer Science from Hunan Normal University, China, in 2002; M.S. in Computer Software and Theory from Central South University, China, in 2008; and his Ph.D. degree from the College of Information Science and Engineering at Hunan University, China, in 2018. He is currently an assistant professor at the College of Applied Mathematics and Computer Engineering, Changsha University, China. His interests are information security, privacy protection, and applied cryptography.

**Zheng Qin** received the Ph.D. degree in computer software and theory from Chongqin University, China, in 2001. He is a professor of computer science and technology in Hunan University, China. He is a member of China Computer Federation (CCF) and ACM respectively. His main interests are computer network and information security, cloud computing, big data processing and software engineering. He has accumulated rich experience in products development and application services, such as in the area of financial, medical, military and education sectors.



**JiXin Zhang** received the B.S. degree from Hubei Polytechnie University in Mathematics, and the M.S. degree from Wuhan University of Technology in computer science and technology, in 2007 and 2014, respectively. Currently, he is pursuing the Ph.D. degree in the College of Information Science and Engineering at Hunan University, China. His research focuses on polymorphic malware detection, privacy protection and big data.

**Lu Ou** received the B.S. degree in computer science from the Changsha University of Science and Technology and the M.S. degree in software engineering from Hunan University, China, in 2009 and 2012, respectively, where she is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering. Her research focuses on security, privacy, and big data.

**Fangmin Li** received the B.Sc. degree from the Huazhong University of Science and Technology, Wuhan, China, in 1990, the M.Sc. degree from the National University of Defense Technology, Changsha, China, in 1997, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2001, all in computer science. He is currently a Professor with the School of Computer Engineering and Applied Mathematics, Changsha University. He has authored several books on embedded systems and over 50 academic papers in wireless networks, and also holds ten Chinese patents. His current research interests include wireless communications and networks security, computer systems and architectures, and embedded systems. Dr. Li is a Senior Member of China Computer Federation (CCF), and a Committee Member of the Technical Committee on Sensor Network, CCF.

**Keqin Li** is a SUNY Distinguished Professor of computer science in the State University of New York. His current research interesting include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 620 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently serving or has served on the editorial boards of IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Transactions on Services Computing, and IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.