# Privacy-preserving Ranked Multi-Keyword Search Leveraging Polynomial Function in Cloud Computing

Yanzhi Ren[1], Yingying Chen[1], Jie Yang[2], Bin Xie[3]
[1]Department of ECE, Stevens Institute of Technology, Hoboken, NJ 07030
*{yren2, yingying.chen}@stevens.edu*
[2]Department of CS, Florida State University, Tallahassee, FL 32306
*jyang5@fsu.edu*
[3]InfoBeyond Technology LLC, Louisville, KY 40223
*Bin.Xie@InfoBeyonds.com*

***Abstract*—The rapid deployment of cloud computing provides users with the ability to outsource their data to public cloud for economic savings and flexibility. To protect data privacy, users have to encrypt the data before outsourcing to the cloud, which makes the data utilization, such as data retrieval, a challenging task. It is thus desirable to enable the search service over encrypted cloud data for supporting effective and efficient data retrieval over a large number of data users and documents in the cloud. Existing approaches on encrypted cloud data search either focus on single keyword search or become inefficient when a large amount of documents are present, and thus have little support for the efficient multi-keyword search. In this paper, we propose a light-weight search approach that supports efficient multi-keyword ranked search in cloud computing system. Specifically, we first propose a basic scheme using polynomial function to hide the encrypted keyword and search patterns for efficient multi-keyword ranked search. To enhance the search privacy, we propose a privacy-preserving scheme which utilizes the secure inner product method for protecting the privacy of the searched multi-keywords. We analyze the privacy guarantee of our proposed scheme and conduct extensive experiments based on the real-world dataset. The experiment results demonstrate that our scheme can enable the encrypted multi-keyword ranked search service with high efficiency in cloud computing.**

## I. Introduction

Cloud computing becomes more and more popular and plays an increasingly important role in our daily lives. In particular, cloud users can remotely outsource their data into the cloud and enjoy the on-demand services from the shared computing resources [1]. Cloud computing brings users with many benefits such as the relief of the storage load and flexible data access, which motivate users to store their local data into the cloud. As the cloud services become prevalent, more and more sensitive information, such as personal photos, government records and finance data, are outsourced into the cloud. To protect the privacy of the sensitive data in the cloud, the data has to be encrypted by the data owner before outsourcing to the cloud [2]. However, data encryption makes effective data utilization a challenging task when a large amount of data files are present: users may have to download the whole data set from the cloud and then decrypt it to conduct keyword search over the data, which is very inefficient when the number of data files is large. Thus, effective keyword searching over encrypted data is of paramount importance, especially need to provide efficient ranked multiple keyword search, which supports a set of input keywords and achieves high efficiency simultaneously in user's search behaviors.

Nevertheless, enabling the keyword search over encrypted data is not an easy task. Some techniques [3]–[5] allow the user to search over encrypted data securely through single keyword to retrieve documents of interest. This is insufficient as many users may tend to provide multiple keywords instead of one as their search interest. Recently, methods have been proposed for multiple keyword search in cloud computing [6], [7]. In these methods, a binary index vector needs to be built for each document and each bit denotes whether the corresponding keyword is included in the document. The storing and updating index can be of substantial overhead, especially when the number of keywords is large. Thus, the efficiency of secure multiple keyword search has large room for improvement for enhancing the system usability in cloud computing.

In this paper, we perform multi-keyword search over encrypted data in clouds leveraging polynomial functions. Specifically, we exploit the number of query keywords appearing in the document index to evaluate the similarity between the query and the document. Our scheme eliminates the predefined binary index vector used in existing multiple keyword search scheme [6] and enables efficient index update, making it scalable to a large number of searching keywords. To meet the challenge of keyword search without privacy leakage, we first propose a multi-keyword search scheme by exploiting the polynomial functions to hide the encrypted keywords. With this approach, the search query can be described as the coefficient vector of polynomial functions which can prevent the adversary from learning the input keywords. To combat the adversary equipped with powerful computation resources, we integrate our polynomial function based approach with the existing secure inner product scheme adapted from the secure k-nearest neighbor (kNN) technique [8]. To validate

the feasibility of our approach, we conducted extensive experiments using real world dataset called Enron Email Dataset. The results show that our scheme is effective and efficient for conducting ranked multiple keyword search.

The rest of the paper is organized as follows. We first present related work in Section II. We then present the system model, threat model and system design goals in Section III. Next, we present our detailed multi-keyword searching scheme in Section IV. In Section V, we evaluate proposed searching scheme using real world dataset. Finally, we conclude our work in Section VI.

## II. Related Work

There have been active studies [3], [4], [9], [10] in designing schemes for keyword search over encrypted data. In these studies, the symmetric key settings are used for keyword search. Keyword search over encrypted data is first studied by Song et al [3] and improvements are provided by Goh et al. [9], Chang et al. [4] and Wang et al. [10]. These methods are developed as crypto primitives and are not able to support high service requirements like searching experience and system usability. Furthermore, these works only allow single keyword search, which is insufficient as many users may provide multiple keywords instead of one as their search input. Thus, it is desirable to seek search schemes that support high service requirements for keyword searching over encrypted data.

Along this direction, some subsequent works [11]–[13] try to solve the conjunctive and disjunctive keyword search. However, they incur large computational overhead since most of them are based on public-key cryptography. An effective multi-keyword ranked search scheme [6], [7] is proposed. In this scheme, users send search requests to the server, who searches over the keyword indexes generated by the data owner. The server then returns a subset of encrypted documents to the user without revealing the keywords in the query and the index. However, in this scheme, a binary vector needs to be built for each document as an index where each bit denotes whether corresponding keyword is included in the document. This means that the user has to know the keyword list and the keywords' positions in the binary vector to generate a query. The index storing and updating can be of substantial overhead, especially when the number of keywords is large. [14]–[17] propose a privacy-preserving multi-keyword text search scheme with similarity-based ranking. However, in [14]–[16], a tree-based index structure is used to improve the search efficiency, and they cannot handle the data updates efficiently: the data owner needs to update the whole search index when inserting or deleting a document. In [17], an additional trusted server is needed to support the index and trapdoor generation.

In our work, we design an efficient secure ranked multiple keyword search by employing polynomial functions to hide the encrypted keywords. Our approach eliminates the requirement of building pre-defined binary index vector in the keyword search, and can handle the data updates efficiently.

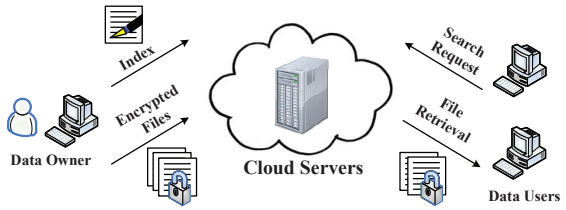


Fig. 1. Architecture of keyword search over encrypted data in cloud computing.

## III. Data Search Model

In this section, we first describe the system model of cloud computing system that is used in this work. We then present threat models we considered in the cloud computing system. We next provide an overview of design goals for the multiple keywords search scheme.

### A. System Model

As illustrated in Figure 1, we consider a cloud data hosting system which consists of three different entities: the data owner, the data user, and the cloud server. The data owner has $n$ data documents $F = (F_1, F_2, ..., F_n)$ that he wants to outsource to the cloud in the encrypted form $C = (C_1, C_2, ..., C_n)$. The data owner is also responsible for building an encrypted searchable index $I$ to enable the keyword searching over $C$. Both the encrypted index $I$ and encrypted files $C$ are then outsourced to the cloud server by the data owner.

We assume that the authorizations have been appropriately done between the data owner and the users. To search the data documents stored in the server, the user will first specify a keyword set $W'$ which consists of $u$ keywords. The user then generates and submits a search request in the secret form (i.e, the trapdoor $T_{W'}$ of keyword set $W'$) to the cloud server. We consider the ranked multi-keyword search problem as follows: upon receiving $T_{W'}$ from the data user, cloud server returns the corresponding encrypted documents after searching the encrypted index $I$. To improve the usability of the system, the cloud server should rank the search results according to certain criteria instead of returning undifferentiated results. Moreover, to reduce the communication overhead, the cloud server only sends back the top $k$ most relevant documents for the user's search request. Finally, the user decrypts the documents received through the access control mechanism.

### B. Threat Model

In the cloud computing system, the system is not maintained by the data owner and thus it is vulnerable to security threats. We consider an adversary which can intercept the network traffic between data user (or data owner) and the server [18]. Specifically, we assume the adversary is curious to infer additional information from the transmission data (i.e., the encrypted data $C$, encrypted index $I$ and trapdoor $T_{W'}$). Based on the information the adversary knows, similar as [6], we consider two threat models for privacy-preserving search in cloud computing system:

**Known Ciphertext Attack.** The cloud adversary knows the encrypted data transmitted between data user (or data owner)

and the server by intercepting the communication. In this case, the adversary may generate new search requests by collecting some valid search requests. Only performing hashing and encryption are unable to prevent this type of attacks due to their deterministic properties.

**Known Background Attack.** In this stronger model, the adversary further possess some background information on the datasets, such as the subject of the dateaset and its statistical information of the keywords. Thus, the adversary can infer keywords by utilizing the captured encrypted keyword frequency and the background information. Therefore, the search pattern is not well protected, while it is a privacy leakage problem in searchable encryption schemes because of the deterministic property of the search request generation.

### C. Design Goals.

In this paper, we design a ranked multi-keyword search scheme to achieve efficient yet privacy-preserving keyword search over encrypted data in cloud computing system. Specially, several goals should be achieved simultaneously by our system design:

**Ranked Multi-keyword Search.** Our search schemes over encrypted data should support multi-keyword query and similarity ranking simultaneously for data retrieval in cloud computing.

**Privacy-preserving.** Our search schemes should meet privacy requirements by preventing the adversary from learning additional information from the data intercepted.

**High Efficiency.** Our proposed functionality and privacy goals should be achieved with low computation and communication cost. Additionally, the encrypted documents $C$ and searchable index $I$ need to be updated efficiently when the document insertion or deletion happens.

## IV. Multiple keywords search

To allow efficient multi-keyword search on the cloud server, the data user must specify a set of keywords and submit the search request to the server. To preserve the privacy, the input keywords should not be exposed to the adversary. In this section, we first propose a search scheme using polynomial function to enable ranked multi-keyword search and hide input keywords in cloud computing system, and then show how to improve it to be privacy-preserving against different levels of threat models in our framework.

### A. Preliminary

*1) Notations:* The data user searches for the data files over documents on the cloud server and we adopt the following notations in this paper:

- $F$: the collection of $n$ plaintext data documents is denoted as:
$$F = \{F_i, i = 1, ..., n\} \quad (1)$$
- $C$: the collection of $n$ encrypted data documents stored in the cloud server is denoted as:
$$C = \{C_i, i = 1, ..., n\} \quad (2)$$
- $W$: $n$ sets of keywords specified by the data owner for the corresponding $n$ data documents are denoted as:
$$W = \{W_i, i = 1, ..., n\} \quad (3)$$
Where each keyword set $W_i$ has $m_i$ keywords and they are denoted as $W_i = \{w_{i,j}, j = 1, ..., m_i\}$.
- $I$ and $\tilde{I}$: the searchable index $I$ ($\tilde{I}$, resp) built for each keyword from $W$ in ranked multi-keyword search scheme and privacy-preserving ranked multi-keyword search scheme. They are denoted as:
$$I = \{I_i, i = 1, ..., n\} \quad (4)$$
$$\tilde{I} = \{\tilde{I}_i, i = 1, ..., n\} \quad (5)$$
Where each subindex $I_i$ ($\tilde{I}_i$, resp) for $W_i$ is denoted as $I_i = \{I_{i,j}, j = 1, ..., m_i\}$ and $\tilde{I}_i = \{\tilde{I}_{i,j}, j = 1, ..., m_i\}$.
- $W'$: the data user specifies $u$ keywords which are denoted as:
$$W' = \{w'_k, k = 1, ..., u\} \quad (6)$$
- $T_{W'}$ and $\tilde{T}_{W'}$: the trapdoor generated for the search query $W'$ in ranked multi-keyword search scheme and privacy-preserving ranked multi-keyword search scheme, respectively.
- $C_{W'}$: the ranked ID list of the top $k$ encrypted data documents returned to the data user for the trapdoor $T_{W'}$ or $\tilde{T}_{W'}$.

*2) Framework:* Followed other works [6], [7], [14] on keyword search over encrypted cloud data, our scheme also contains four major procedures: Setup, BuildIndex, TrapDoor and Query:

- **Setup:** The data owner randomly generates a secret key $SK$ and distributes it to the authorized data users.
- **BuildIndex:** Based on the keyword set $W$, the data owner generates the searchable index $I$ which is encrypted by the key $SK$. The owner then encrypts the plaintext data collection $F$ into encrypted data collection $C$ and publishes the index $I$ and $C$ to the cloud server.
- **TrapDoor:** The data user generates a secure trapdoor $T_{W'}$ corresponding to his/her input keyword set $W'$ using the key $SK$.
- **Query:** When the cloud server receives the search query $T_{W'}$, it performs the keyword search on the index $I$ using $T_{W'}$ and returns the ranked ID list of top $k$ documents $C_{W'}$ to the data user.

### B. Polynomial Function Based Ranked Multi-keyword Search

*1) Search Scheme:* To provide a guarantee against violation on privacy and security requirements, we have to hide the input keyword set $W'$ in the search query. To do so, we encrypt these keywords and then construct a polynomial function to hide them in search trapdoor generation. The detailed scheme to achieve the ranked multi-keyword search over encrypted data is as follows:

**Setup:** The data owner generates an encryption function $E()$ and a hash function $H()$ to form the secret key as

$SK = \{E(), H()\}$. The data owner then sends $SK$ to authorized users.

**BuildIndex:** The data owner extracts $m_i$ keywords $\{w_{i,j}, j = 1, ..., m_i\}$ from each data document (i.e., document $F_i$) to build the search index. To prevent the adversary from learning the index keywords, the data owner encrypts each keyword with the key $SK$. The encrypted keywords for document $F_i$ are denoted as $\{H(E(w_{i,j})), j = 1, ..., m_i\}$. The data owner then computes different powers of each encrypted keyword and builds its search index as: $I_{i,j} = \left((H(E(w_{i,j})))^0, ..., (H(E(w_{i,j})))^d\right)^T$ ($d$ is the maximum number of input keywords and we will describe it later). Further, it creates an matrix to hold the index information for $m_i$ keywords of document $F_i$ as:

$$\begin{aligned} I_i &= (I_{i,1}, ..., I_{i,m_i}) \\ &= \begin{pmatrix} (H(E(w_{i,1})))^0 & \dots & (H(E(w_{i,m_i})))^0 \\ \vdots & \ddots & \vdots \\ (H(E(w_{i,1})))^d & \cdots & (H(E(w_{i,m_i})))^d \end{pmatrix} \end{aligned} \quad (7)$$

The data owner then publishes encrypted data $C$ and index $I = \{I_i, i = 1, ..., n\}$ for $n$ data documents to the cloud server.

**TrapDoor:** The data user specifies a keyword set $W' = \{w'_1, ..., w'_u\}$ which consists of $u$ keywords as the search input. Let $d$ be the maximum number of input keywords allowed in the cloud computing system. To make the number of keywords consistent, the user then adds $d - u$ dummy keywords $\{w'_{u+1}, ..., w'_d\}$ to the set $W'$ to make sure the total number of input keywords is $d$:

$$W' = \{w'_1, ..., w'_u, w'_{u+1}, ..., w'_d\} \quad (8)$$

We note that each dummy keyword consists of a mixed sequence of randomly generated characters and numbers and it is different from any real dictionary words. Thus, these dummy words will not impact the search results. The data user then encrypts these $d$ keywords by using the key $SK = \{E(), H()\}$ received from the data owner:

$$H(E(W')) = \{H(E(w'_k)), k = 1, ..., d\} \quad (9)$$

Provided with the knowledge about the roots of polynomials, we know that the factorization of polynomials is very computationally intense, especially when the order of polynomials becomes large [19]. With the aid of such clues, in this work, we utilize the polynomial functions to hide the encrypted input keywords. Specifically, the data user constructs a polynomial function of degree $d$ as:

$$\begin{aligned} f(x) &= (x - H(E(w'_1))) \times ... \times (x - H(E(w'_d))) \\ &= b_0 + b_1 x + ... + b_d x^d \end{aligned} \quad (10)$$

It is obvious that the polynomial function meets the requirement that $f(H(E(w))) = 0$ if and only if the keyword $w \in W'$. The user thus utilizes the coefficients of polynomial function to form a search request and sends them to the cloud server:

$$T_{W'} = \{b_0, ..., b_d\}^T \quad (11)$$

**Query:** With the help of the trapdoor $T_{W'}$ and the index $I_i$ built for each data document $F_i$, cloud server computes the $Y_i$ as:

$$\begin{aligned} Y_i &= (T_{W'})^T I_i \\ &= (T_{W'})^T (I_{i,1}, ..., I_{i,m_i}) \\ &= (b_0, ..., b_d) \begin{pmatrix} (H(E(w_{i,1})))^0 & \dots & (H(E(w_{i,m_i})))^0 \\ \vdots & \ddots & \vdots \\ (H(E(w_{i,1})))^d & \cdots & (H(E(w_{i,m_i})))^d \end{pmatrix} \\ &= (Y_i(1), ..., Y_i(m_i)) \end{aligned} \quad (12)$$

From the previous analysis, we know that if $w_{i,j} \in W'$, we have $Y_i(j) = (T_{W'})^T I_{i,j} = 0$. We define the similarity score between the search query $T'_W$ and index $I_i$ as the number of word matches between them. Thus, the number of zero values in vector $Y_i$ can be regarded as the similarity score between the search query $T'_W$ and index $I_i$. The server then computes similarity scores between search query and each $I_i$ $(i = 1, .., n)$ by counting the number of zero values in the corresponding $Y_i$. After sorting all these similarity scores, cloud server returns the top $k$ ranked ID list $C_{W'}$ to the user, in which $k$ denotes the number of documents returned to the user for his/her search request.

*2) Analysis:* From the descriptions above, the final similarity score is defined as the number of query keywords appearing in a data document index $I_i$. Therefore the order of similarity is preserved for all the data documents. When the server returns the top $k$ ranked ID list $C_{W'}$ to the user, the most similar documents to the query are included with clear order. Additionally, our scheme eliminates the pre-defined binary index vector built for keywords: when a new keyword is added in the search index, the data owner only needs to compute different powers of this encrypted keyword and adds one new column in the search index as shown in Equation 7. Thus, our scheme can handle dynamic data updates efficiently. Our scheme also satisfies the privacy guarantee of searchable encryption schemes as described below.

**Known Ciphertext Attack:** In this threat model, the adversary can intercept the encrypted document collection $C$, the index $I$ and the query $T_{W'}$. In this case, it is computationally intensive for the adversary to conduct the factorization of the polynomial function used in query $T_{W'}$ and guess the encrypted keywords in $H(E(W'))$. Thus, the adversary is not able to generate new search request by collecting valid search request. Additionally, the keywords in the query and index are also encrypted: their privacy is also protected as long as the secret key $SK = \{E(), H()\}$ is kept confidential. Thus, our multiple keyword search scheme is secure against this threat model.

**Known Background Attack:** In this threat mode, the adversary intends to deduce keywords from the search frequency by using his background information on the dataset. One uniqueness of our multiple keyword search scheme is that it can generate two different query data for the same set of keywords $W'$ because of the randomly generated dummy keywords.

Therefore, the query is not generated in a deterministic manner and the adversary is not able to tell the search frequency of any keywords. Consequently, the adversary cannot deduce keywords due to the lack of keyword search frequency. Thus, our multiple keyword search scheme is also secure against the known background attack.

### C. Integration with Secure Inner Product

Our proposed ranked multiple keyword search scheme hides the keywords in the search query using polynomial functions, which provides some the privacy guarantees over two threat models. However, it still incurs some privacy leakages: the adversary can deduce the encrypted keywords by conducting factorization of the polynomial function in the query $T_{W'}$ given enough computation resources of the adversary. In addition, the adversary can also get to know the encrypted keywords from the search index $I$. Therefore, the adversary may generate a new valid trapdoor or know the search patterns using these deduced encrypted keywords. In this part, we propose a more advanced scheme called privacy-preserving ranked multi-keyword search to be privacy preserving under such advanced adversaries.

*1) Search Scheme:* In our proposed ranked multi-keyword search scheme, we compute the inner product of the trapdoor $T_{W'}$ and each $I_{i,j}$ in index $I_i$, and the number of zero values in the result $Y_i$ is regarded as the number of query keywords appearing in the corresponding document, which is used to evaluate the similarity between them. However, the previous analysis shows that the encrypted keywords in the trapdoor $T_{W'}$ can be deduced by the adversary who is able to do factorization of the polynomial function in the query $T_{W'}$. Therefore, it is necessary to further hide the inner product of the trapdoor and search index to protect the privacy. In [8], the secure k-nearest neighbor (kNN) technique has been proposed. Similar as [6], [14], we tailor the secure kNN scheme as the secure inner product method and propose privacy-preserving ranked multi-keyword ranked search scheme. The details of the improved search scheme are shown as follows.

**Setup:** To improve the security of our keyword search scheme, we further encrypt the trapdoor and index using some randomly generated matrices and vectors. Thus, besides the encryption function $E()$ and hash function $H()$ generated in ranked multi-keyword search scheme, the data owner further generates: (1) Two random $d \times d$ invertible matrices $M_1$ and $M_2$; (2) A random binary string $S$ of $d$ bits. We use $S(k)$ to denote the $k$-th bit in $S$. The data owner then forms the key $SK = \{E(), H(), M_1, M_2, S\}$ and sends it to authorized users.

**BuildIndex:** To provide a guarantee against privacy violation, we further encrypt the search index by utilizing the parameters $\{M_1, M_2, S\}$ generated in the setup procedure. Consider each sub-index $I_{i,j}$ built for keyword $w_{i,j}$ in our ranked multi-keyword search: $I_{i,j} = \left((H(E(w_{i,j})))^0, ..., (H(E(w_{i,j})))^d\right)^T$: (1) The data owner splits $I_{i,j}$ into two random vectors $I^a_{i,j}$ and $I^b_{i,j}$: for $k = 1$ to $d$, if $S(k) = 1$, the data owner randomly divides $I_{i,j}(k)$ into $I^a_{i,j}(k)$ and $I^b_{i,j}(k)$ so that $I^a_{i,j}(k) + I^b_{i,j}(k) = I_{i,j}(k)$. If $S(k) = 0$, the data owner sets both $I^a_{i,j}(k)$ and $I^b_{i,j}(k)$ to $I_{i,j}(k)$. (2) The data owner then encrypts $I_{i,j}$ as $M_1^T I^a_{i,j}$ and $M_2^T I^b_{i,j}$. Thus, the index for each keyword $w_{i,j}$ is built as $\tilde{I}_{i,j} = \{M_1^T I^a_{i,j}, M_2^T I^b_{i,j}\}$ in our privacy-preserving ranked multi-keyword search scheme.

**TrapDoor:** To protect the privacy, we have to eliminate the relationships in search query $T_{W'}$ so that the adversary is not able to deduce the encrypted keywords using factorization of the polynomial function. To do so, we utilize the parameters $\{M_1, M_2, S\}$ to further encrypt the query $T_{W'}$ used in ranked multi-keyword search scheme.

Consider the query $T_{W'} = \{b_0, ..., b_d\}^T$ built in ranked multi-keyword search: (1) The data user then splits $T_{W'}$ into two random vectors $T^a_{W'}$ and $T^b_{W'}$ with similar procedures. The difference is that for $k = 1$ to $d$, if $S(k) = 0$, the data user randomly divides $T_{W'}(k)$ into $T^a_{W'}(k)$ and $T^b_{W'}(k)$ so that $T^a_{W'}(k) + T^b_{W'}(k) = T_{W'}(k)$. If $S(k) = 1$, the data user sets both $T^a_{W'}(k)$ and $T^b_{W'}(k)$ to $T_{W'}(k)$. (2) The data user then encrypts $T_{W'}$ as $M_1^{-1} T^a_{W'}$ and $M_2^{-1} T^b_{W'}$ to generate the trapdoor as $\tilde{T}_{W'} = \{M_1^{-1} T^a_{W'}, M_2^{-1} T^b_{W'}\}$.

**Query:** With the trapdoor $\tilde{T}_{W'}$, cloud server searches through each sub-index $I_{i,j}$ for keyword $w_{i,j}$ and determines whether it is included in the trapdoor by computing $\tilde{Y}_i(j)$ using the trapdoor $\tilde{T}_{W'}$ and index $\tilde{I}_{i,j}$.

$$\begin{aligned}
\tilde{Y}_i(j) &= (\tilde{T}_{W'})^T \cdot \tilde{I}_{i,j} \\
&= \{M_1^{-1} T^a_{W'}, M_2^{-1} T^b_{W'}\}^T \cdot \{M_1^T I^a_{i,j}, M_2^T I^b_{i,j}\} \\
&= (T^a_{W'})^T (M_1^{-1})^T M_1^T I^a_{i,j} + (T^b_{W'})^T (M_2^{-1})^T M_2^T I^b_{i,j} \\
&= (T^a_{W'})^T I^a_{i,j} + (T^b_{W'})^T I^b_{i,j} \\
&= (T_{W'})^T \cdot I_{i,j} \\
&= Y_i(j)
\end{aligned} \tag{13}$$

Thus, we have $\tilde{Y}_i(j) = 0$ if $w_{i,j} \in W'$ and the number of zero values in $\tilde{Y}_i = (\tilde{Y}_i(1), ..., \tilde{Y}_i(m_i))$ can be used as the indicator for the number of word matches between the query $\tilde{T}_{W'}$ and index $\tilde{I}_i$. Similarly, we compute the similarity scores between the search query and each $\tilde{I}_i$ using its corresponding $\tilde{Y}_i$ $(i = 1, .., n)$. Finally, the cloud server returns the top $k$ ranked ID list $C_{W'}$ back to the data user after sorting all scores.

*2) Analysis:* The privacy-preserving ranked multi-keyword search scheme can help to solve the privacy problems existed in ranked multi-keyword scheme. The index and query vectors are first randomly splitted into two vectors and encrypted using the random matrices $M_1$ and $M_2$ in BuildIndex and TrapDoor procedure. Thus, in privacy-preserving scheme, the adversary is not able to deduce the encrypted keywords in the trapdoor and the search index. The trapdoor and index privacy is well protected by the secure inner product scheme as long as the parameters $\{M_1, M_2, S\}$ are kept confidential [8]. In addition, our scheme can also generate two totally different trapdoors for the same query $W'$ because of the random dummy keywords. Therefore, the search pattern is also well protected.

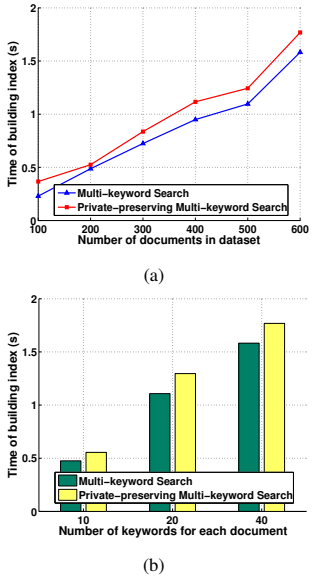


Fig. 2. Time cost on index construction: (a) Under different number of documents in dataset when the number of keywords extracted for each document is set as 40. (b) Under different number of keywords extracted for each document when the number of documents in dataset is 600.

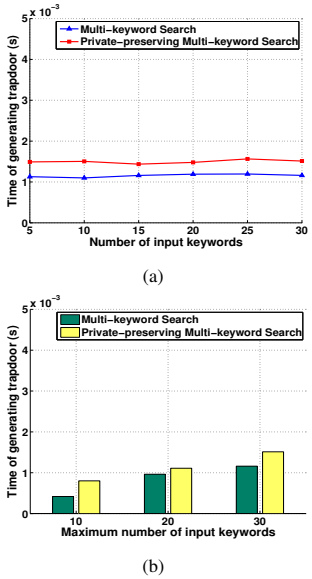


Fig. 3. Time cost on trapdoor generation: (a) Under different number of keywords input by the user when the maximum number of input keywords is 30. (b) Under different maximum number of input keywords when the number of keywords input by the user is 10.

## V. Performance Evaluation

In this section, we first present our experimental methodology for evaluating the proposed multiple keyword search schemes. We then show the experimental results to demonstrate the effectiveness and efficiency of our proposed search schemes.

### A. Experimental Methodology

We use the real world dataset called Enron Email Dataset which contains emails and messages from about 150 users [20] to evaluate our approach. Different number of emails are randomly selected from the Enron Email to build our experimental dataset. For each set of input keywords randomly generated by the user, the cloud server will search through the dataset and retrieve the qualified files. We implement the whole system on a computer with a 2.40GHz Core 2 P8600 Processor and 2G DDR2 memory. To get the statistical results of our approach, we repeat the simulations for 100 times. We use the following metrics to evaluate our proposed schemes.

- **Time Cost on Index Construction.** The average time for the data owner to build the search index, which consists of extracting and encrypting the keyword set for each data document.
- **Time Cost on Trapdoor Generation.** The average time for the data user to prepare a search trapdoor, which consists of encrypting the input keywords and generating the search query.
- **Time Cost on Query.** The average time for the cloud server to accomplish a search request, which consists of computing and ranking the similarity scores for the data documents.

### B. Index Construction

In the first set of experiments, we evaluate the time cost on building the search index using our two proposed schemes: the ranked multi-keyword search scheme and the private-preserving ranked multi-keyword search scheme. Specifically, we first vary the number of documents in dataset from 100 to 600, with 40 keywords extracted from each document (i.e., $m_i = 40$ for $i = 1, ..., n$). As shown in Figure 2 (a), we observe that time cost of building the search index increases as the number of documents increases. Further, we find that the time cost is almost linear with the number of documents in dataset due to the time cost for building a sub-index for each document is almost fixed. Moreover, our basic ranked multi-keyword search scheme achieves a lower time cost than privacy-preserving version under each number of documents. This observation is also inline with our analysis because privacy-preserving scheme further encrypts the index using two randomly generated matrices.

We then vary the number of keywords extracted from each document (i.e., $m_i$) to evaluate the efficiency of our scheme in Figure 2 (b). We observe that the larger number of keywords of each document results in higher time cost for building the search index. This is because more sub-indexes need to be built for larger number of keywords. Again, as we have observed previously, the ranked multi-keyword search scheme achieves a lower time cost than that of the privacy-preserving version under different number of keywords.

### C. Trapdoor Generation

We next evaluate the time cost on trapdoor generation of our search schemes. Figure 3 (a) presents the time cost of trapdoor generation under different number of input keywords when the maximum number of input keywords is 30 (i.e., $d = 30$). We observe that the number of input keywords specified by the user has little influence on the time cost of trapdoor generation and the overall time cost is below 0.002 seconds. This is because dummy words are added to the input keyword set to make sure the total number of keywords is $d$, no

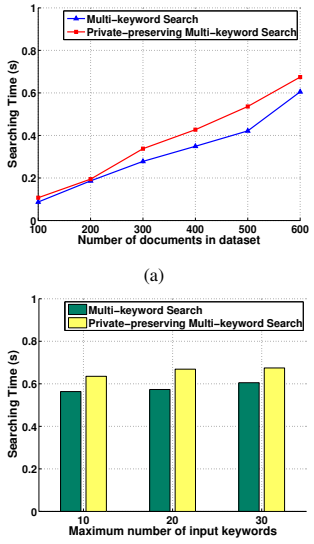


Fig. 4. Time cost on query: (a) Under different number of documents in dataset when the maximum number of input keywords is 30. (b) Under different maximum number of input keywords when the number of documents is 600.

matter what the number of keywords input by the data user is. Figure 3 (b) demonstrates the time cost of trapdoor generation under different maximum number of input keywords (i.e., $d$). We observe that the time cost increases as the number of keywords increases. This is because when the number of keywords becomes larger, the order of the polynomial function used for trapdoor generation also increases. Again, we observe that the ranked multi-keyword search scheme achieves a lower time cost than that of privacy-preserving version, which is consistent with our previous observations.

### D. Query

Finally, we evaluate the effectiveness of our search scheme in terms of the time cost on query. We first keep the maximum number of input keywords as 30 (i.e., $d = 30$) and the number of keywords extracted for each data document as 40 (i.e., $m_i = 40$ for $i = 1, ..., n$) while varying the number of documents in the dataset. Figure 4 (a) shows that the query time increases as the number of documents increases. The query time is dominated by the number of documents. This is because with more documents, it takes longer time for the server to search over the dataset. We then keep the number of documents as 600 while varying the maximum number of input keywords (i.e., $d$). Figure 4 (b) shows that the similar query time is achieved even if the maximum number of keywords in the query varies, indicating our scheme is not sensitive to the maximum number of input keywords.

In summary, our experimental results indicate that our schemes, both ranked multi-keyword search scheme and privacy-preserving ranked multi-keyword search scheme, can enable the multi-keyword searching with high efficiency in cloud computing.

## VI. Conclusion

In this paper, we propose a light-weight search approach that supports efficient multi-keyword ranked search in cloud computing system. Our basic scheme employs the polynomial function to hide the encrypted keyword and search patterns for efficient multi-keyword ranked search. We then improve the basic scheme and propose a privacy-preserving scheme which utilizes the secure inner product method for protecting the privacy of the searched multi-keywords. Thorough analysis on the privacy guarantee of our proposed schemes is given, and extensive experiments based on the real-world dataset are also conducted. The experiment results demonstrate that our scheme can enable the encrypted multi-keyword ranked search service with high efficiency in cloud computing.

## References

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, 2008.
[2] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proceedings of the 14th international conference on Financial cryptograpy and data security*, 2010.
[3] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, 2000.
[4] Y.-C. e. a. Chang, "Privacy preserving keyword searches on remote encrypted data," in *Proceedings of ACNS*, 2005.
[5] M. Abdalla and et al., "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," *J. Cryptol.*, 2008.
[6] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proceedings of IEEE INFOCOM*, 2011.
[7] N. Cao and et al., "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, Jan 2014.
[8] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the ACM SIGMOD International Conference on Management of data*, 2009.
[9] E.-J. Goh, "Secure indexes," *Cryptology ePrint Archive*, 2003.
[10] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proceedings of ICDCS*, 2010.
[11] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proceedings of International Conference on Pairing-Based Cryptography*, 2007.
[12] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proceedings of 2nd International Conference on Applied Cryptography and Network Security*, 2004.
[13] S. Hou and et al., "Privacy preserving confidential forensic investigation for shared or remote servers," in *Proceedings of IIH-MSP*, 2011.
[14] W. Sun and et al., "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proceedings of ACM SIGSAC*, 2013.
[15] 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," *IEEE Transactions on Parallel and Distributed Systems*, 2014.
[16] B. Wang, Y. Hou, M. Li, H. Wang, and H. Li, "Maple: Scalable multi-dimensional range search over encrypted cloud data with tree-based index," in *Proceedings of ACM ASIACCS*, 2014.
[17] S. Zittrower and C. Zou, "Encrypted phrase searching in the cloud," in *Proceedings of GLOBECOM*, 2012.
[18] R. L. Krutz and R. D. Vines, *Cloud Security: A Comprehensive Guide to Secure Cloud Computing*. Wiley, 2010.
[19] D. Starer and A. Nehorai, "High-order polynomial root tracking algorithm," in *Proceedings of the IEEE ICASSP*, 1992.
[20] W.W.Cohen, "Enron email dataset," 2009. [Online]. Available: www.cs.cmu.edu/~enron/