

A protected and active Multi-keyword Ranked Search intend over Encrypted Cloud Records

Mr. T.Charan Singh¹, Mrs. K. Jhansi², Mrs. K. Praveena²

¹Asst.Professor, Sriindu College of Engineering & Technology, Hyderabad

²P.G. Scholar, Sriindu College of Engineering & Technology, Hyderabad

Abstract

With the advent of cloud computing, data owners are motivated to outsource their complex data management systems from local sites to the commercial public cloud for great flexibility and economic savings. But for protecting data privacy, sensitive data has to be encrypted before outsourcing, which obsoletes traditional data utilization based on plaintext keyword search. Thus, enabling an encrypted cloud data search service is of paramount importance. Considering the large number of data users and documents in the cloud, it is necessary to allow multiple keywords in the search request and return documents in the order of their relevance to these keywords. Related works on searchable encryption focus on single keyword search or Boolean keyword search, and rarely sort the search results. In this paper, for the first time, we define and solve the challenging problem of privacy- preserving multi-keyword ranked search over encrypted cloud data (MRSE). We establish a set of strict privacy requirements for such a secure cloud data utilization system. Among various multi- keyword semantics, we choose the efficient similarity measure of “coordinate matching”, i.e., as many matches as possible, to capture the relevance of data documents to the search query. We further use “inner product similarity” to quantitatively evaluate such similarity measure. We first propose a basic idea for the MRSE based on secure inner product computation, and then give two significantly improved MRSE schemes to achieve various stringent privacy requirements in two different threat models. Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given. Experiments on the real-world dataset further show proposed schemes indeed introduce low overhead on computation and communication.

INTRODUCTION

Cloud computing is the long dreamed vision of computing as a utility, where cloud customers can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources [1]. Its great flexibility and economic savings are motivating both individuals and enterprises to outsource their local complex data management system into the cloud. To protect data privacy and combat unsolicited accesses in the cloud and beyond, sensitive data, e.g., emails, personal health records, photo albums, tax documents, financial transactions, etc.,

We have to be encrypted by data owners before outsourcing to the commercial public cloud [2]; this, however, obsoletes the traditional data utilization service based on plaintext keyword search. The trivial solution of downloading all the data and decrypting locally is clearly impractical, due to the huge amount of bandwidth cost in cloud scale systems. Moreover, aside from eliminating the local storage management, storing data into the cloud serves no purpose unless they can be easily searched and utilized. Thus, exploring privacy-

preserving and effective search service over encrypted cloud data is of paramount importance. Considering the potentially large number of on-demand data users and huge amount of outsourced data documents in the cloud, this problem is particularly challenging as it is extremely difficult to meet also the requirements of performance, system usability and scalability. On the one hand, to meet the effective data retrieval need, the large amount of documents demand the cloud server to perform result relevance ranking, instead of returning undifferentiated results. Such ranked search system enables data users to find the most relevant information quickly, rather than burdensomely sorting through every match in the content collection [3]. Ranked search can also elegantly eliminate unnecessary network traffic by sending back only the most relevant data, which is highly desirable in the “pay-as-you-use” cloud paradigm. For privacy protection, such ranking operation, however, should not leak any keyword related information. On the other hand, to improve the search result accuracy as well as to enhance the user searching experience, it is also necessary for such ranking system to support multiple keywords search, as single keyword search often yields far too coarse results.

As a common practice indicated by today's web search engines (e.g., Google search), data users may tend to provide a set of keywords instead of only one as the indicator of their search interest to retrieve the most relevant data. And each keyword in the search request is able to help narrow down the search result further. "Coordinate matching" [4], i.e., as many matches as possible, is an efficient similarity measure among such multi-keyword semantics to refine the result relevance, and has been widely used in the plaintext information retrieval (IR) community. However, how to apply it in the encrypted cloud data search system remains a very challenging task because of inherent security and privacy obstacles, including various strict requirements like the data privacy, the index privacy, the keyword privacy, and many others (see section III-B).

In the literature, searchable encryption [5]–[13] is a helpful technique that treats encrypted data as documents and allows a user to securely search through a single keyword and retrieve documents of interest. However, direct application of these approaches to the secure large scale cloud data utilization system would not be necessarily suitable such high service-level requirements like system usability, user searching experience, and easy information discovery. Although some recent designs have been proposed to support Boolean keyword search [14]–[21] as an attempt to enrich the search flexibility, they are still not adequate to provide users with acceptable result ranking functionality (see section VI). Our early work [22] has been aware of this problem, and provided a solution to the secure ranked search over encrypted data problem but only for queries consisting of a single keyword. How to design an efficient encrypted data search mechanism that supports multi-keyword semantics without privacy breaches still remains a challenging open problem.

In this paper, for the first time, we define and solve the problem of multi-keyword ranked search over encrypted cloud data (MRSE) while preserving strict system-wise privacy in the cloud computing paradigm. Among various multi-keyword semantics, we choose the efficient similarity measure of "coordinate matching", i.e., as many matches as possible, to capture the relevance of data documents to the search query. Specifically, we use "inner product similarity" [4], i.e., the number of query keywords appearing in a document, to quantitatively evaluate such similarity measure of that document to the search query. During the index construction, each document is associated with a binary vector as a subindex where each bit represents whether corresponding keyword is contained in the document. The search query is also described as a binary vector where each bit means whether corresponding keyword appears in this search request, so the similarity could be exactly measured

by the inner product of the query vector with the data vector. However, directly outsourcing the data vector or the query vector will violate the index privacy or the search privacy. To meet the challenge of supporting such multi-keyword semantic without privacy breaches, we propose a basic idea for the MRSE using secure inner product computation, which is adapted from a secure k -nearest neighbor (kNN) technique [4], and then give two significantly improved MRSE schemes in a step-by-step manner to achieve various stringent privacy requirements in two threat models with increased attack capabilities. Our contributions are summarized as follows

- 1) For the first time, we explore the problem of multi-keyword ranked search over encrypted cloud data, and establish a set of strict privacy requirements for such a secure cloud data utilization system.
- 2) We propose two MRSE schemes based on the similarity measure of "coordinate matching" while meeting different privacy requirements in two different threat models.
- 3) Thorough analysis investigating privacy and efficiency guarantees of the proposed schemes is given, and experiments on the real-world dataset further show the proposed schemes indeed introduce low overhead on computation and communication

The remainder of this paper is organized as follows. In Section II, we introduce the system model, the threat model, our design goals, and the preliminary. Section III describes



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

II. PROBLEM FORMULATION

A. System Model

Considering a cloud data hosting service involving three different entities, as illustrated in Fig. 1: the data owner, the data user, and the cloud server. The data owner has a collection of data documents $\mathcal{D}$ to be outsourced to the cloud server in the encrypted form $\mathcal{E}$. To enable the searching capability over $\mathcal{E}$ for effective data utilization, the data owner, before outsourcing, will first build an encrypted searchable index $\mathcal{I}$ from $\mathcal{D}$, and then outsource both the index $\mathcal{I}$ and the encrypted document collection $\mathcal{E}$ to the cloud server. To search the document collection for

given keywords, an authorized user acquires a corresponding trapdoor τ through search control mechanisms, e.g., broadcast encryption [8]. Upon receiving

B. Threat Model

The cloud server is considered as “honest-but-curious” in our model, which is consistent with related works on cloud security [23], [24]. Specifically, the cloud server acts in an “honest” fashion and correctly follows the designated protocol specification. However, it is “curious” to infer and analyze data (including index) in its storage and message flows received during the protocol so as to learn additional information. Based on what information the cloud server knows, we consider two threat models with different attack capabilities as follows the MRSE framework and privacy requirements, followed by

section IV, which describes the proposed schemes. Section V presents simulation results. We discuss related work on both single and Boolean keyword searchable encryption in Section VI, and conclude the paper in Section VII.

from a data user, the cloud server is responsible to search the index $\mathcal{I}$ and return the corresponding set of encrypted documents. To improve the document retrieval accuracy, the search result should be ranked by the cloud server according to some ranking criteria (e.g., coordinate matching, as will be introduced shortly). Moreover, to reduce the communication cost, the data user may send an optional number n along with the trapdoor τ so that the cloud server only sends back top- n documents that are most relevant to the search query. Finally, the access control mechanism [23] is employed to manage decryption capabilities given to users.

Known Ciphertext Model In this model, the cloud server is supposed to only know encrypted dataset $\mathcal{D}$ and searchable index $\mathcal{I}$, both of which are outsourced from the data owner.

Known Background Model In this stronger model, the cloud server is supposed to possess more knowledge than what can be accessed in the known ciphertext model. Such information may include the correlation relationship of given search requests (trapdoors), as well as the dataset related statistical information. As an instance of possible attacks in this case, the cloud server could use the known trapdoor information combined with document/keyword frequency [25] to deduce/identify

C. Design Goals

To enable ranked search for effective utilization of out-sourced cloud data under the aforementioned

model, our system design should simultaneously achieve security and performance guarantees as follows.

- **Multi-keyword Ranked Search:** To design search schemes which allow multi-keyword query and provide result similarity ranking for effective data retrieval, instead of returning undifferentiated results.

- **Privacy-Preserving:** To prevent the cloud server from learning additional information from the dataset

D. Notations

- $\mathcal{X}$ – the plaintext document collection, denoted as a set of data documents $\mathcal{X} = (\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_n)$.

- $\mathcal{Y}$ – the encrypted document collection stored in the cloud server, denoted as $\mathcal{Y} = (\mathcal{Y}_1, \mathcal{Y}_2, \dots, \mathcal{Y}_n)$.

- $\mathcal{K}$ – the dictionary, i.e., the keyword set consisting of $\mathcal{K}$ keyword, denoted as $\mathcal{K} = (\mathcal{K}_1, \mathcal{K}_2, \dots, \mathcal{K}_m)$.

- $\mathcal{I}$ – the searchable index associated with $\mathcal{K}$, denoted as

E. Preliminary on Coordinate Matching

As a hybrid of conjunctive search and disjunctive search, “coordinate matching” [4] is an intermediate similarity measure which uses the number of query keywords appearing in the document to quantify the relevance of that document to the query. When users know the exact subset of the dataset to be retrieved, Boolean queries perform well with the precise search requirement specified by the user. In cloud computing, however, this is not the practical case, given the huge amount of outsourced data. Therefore, it is more flexible for users to certain keywords in the query specify a list of keywords indicating their interest and retrieve the most relevant documents with a rank order.

III. FRAMEWORK AND PRIVACY REQUIREMENTS FOR MRSE

In this section, we define the framework of multi-keyword ranked search over encrypted cloud data (MRSE) and establish various strict system-wise privacy requirements for such a secure cloud data utilization system.

A. MRSE Framework

For easy presentation, operations on the data documents are not shown in the framework since the data owner could easily employ the traditional symmetric key cryptography to encrypt and then outsource data. With focus on the index and query, the MRSE system consists of four algorithms as follows.

- **Setup($\mathcal{K}$)** Taking a security parameter ℓ as input, the data owner outputs a symmetric key as $\mathcal{K}$.

- $\text{BuildIndex}(\mathcal{D}, \mathbb{K})$ Based on the dataset $\mathcal{D}$, the owner builds a searchable index $\mathcal{I}$ which is encrypted by the symmetric key $\mathbb{K}$ and then outsourced to the cloud server. After the index construction, the document collection can be independently encrypted and outsourced.

- $\text{Trapdoor}(\widetilde{Q})$ With $\mathbb{K}$ keywords of interest in $\widetilde{Q}$ as input

Keyword Privacy As users usually prefer to keep their search from being exposed to others like the cloud server, the most important concern is to hide what they are searching, i.e., the keywords indicated by the corresponding trapdoor. Although the trapdoor can be generated in a cryptographic way to protect the query keywords, the cloud server could do some statistical analysis over the search result to make an estimate. As a kind of statistical information, *document frequency* (i.e., the number of documents containing the keyword) is sufficient to identify the keyword with high probability [26]. When the cloud server knows some background information of the dataset, this keyword specific information may be utilized to reverse-engineer the keyword.

Trapdoor Unlinkability The trapdoor generation function should be a randomized one instead of being deterministic. In particular, the cloud server should not be able to deduce the relationship of any given trapdoors, e.g., to determine whether the two trapdoors are formed by the same search request. Otherwise, the deterministic trapdoor generation would give the cloud server advantage to accumulate frequencies of different search requests regarding different keyword(s), which may further violate the aforementioned keyword privacy requirement. So the fundamental protection for trapdoor unlinkability is to introduce sufficient nondeterminacy into the **Access Pattern** Within the ranked search, the access pattern is the sequence of search results where every search result is a set

IV. PRIVACY-PRESERVING AND EFFICIENT MRSE

To efficiently achieve multi-keyword ranked search, we propose to employ “inner product similarity” [4] to quantitatively evaluate the efficient similarity measure “coordinate matching”. Specifically, $\mathbb{D}$ is a binary data vector for document

$\mathbb{D}$ where each bit $\mathbb{D}[\#] \in \{0, 1\}$ represents the existence

of the corresponding keyword $\mathbb{D}$ in that document, and $\mathbb{Q}$

is a binary query vector indicating the keywords of interest

where each bit $\mathbb{Q}[\#] \in \{0, 1\}$ represents the existence of the corresponding keyword $\mathbb{D}$ in the query $\widetilde{Q}$.

The similarity score of document $\mathbb{D}$ to query $\widetilde{Q}$ is therefore expressed as the inner product of their binary column vectors, i.e., $\mathbb{D} \cdot \mathbb{Q}$. For the purpose of ranking, the cloud server must be given the capability to compare the similarity of different documents to the query. But, to preserve strict system-wise privacy, data vector $\mathbb{D}$, query vector $\mathbb{Q}$ and their inner product $\mathbb{D} \cdot \mathbb{Q}$ should not be exposed to the cloud server. In this section, we first propose a basic idea for the MRSE using secure inner product computation, which is adapted from a secure $\#$ -nearest neighbor (kNN) technique, and then show how to significantly improve it to be privacy-preserving against different threat models in the MRSE framework in a step-by-step manner.

A. MRSE_I: Privacy-Preserving Scheme in Known Ciphertext

1) *Secure kNN Computation*: In the secure $\#$ -nearest neighbor (kNN) scheme [28], Euclidean distance between a database record $\mathbb{D}$ and a query vector $\mathbb{Q}$ is used to select nearest database records. The secret key is composed of one $(\# + 1)$ -bit vector as $\mathbb{R}$ and two $(\# + 1) \times (\# + 1)$ invertible matrices as $\{\mathbb{M}_1, \mathbb{M}_2\}$, where $\#$ is the number of fields for each record

$\mathbb{D}$. First, every data vector $\mathbb{D}$ and query vector $\mathbb{Q}$ are extended

to $(\# + 1)$ -dimension vectors as $\widetilde{\mathbb{D}}$ and $\widetilde{\mathbb{Q}}$, where the $(\# + 1)$ -th dimension is set to $-0.5\|\mathbb{D}\|^2$ and 1, respectively. Besides,

the query vector $\widetilde{\mathbb{Q}}$ is scaled by a random number ϕ $\phi > 0$ as $(\phi\widetilde{\mathbb{Q}}, \mathbb{R})$. Then, $\widetilde{\mathbb{D}}$ is split into two random

vectors as $\{\widetilde{\mathbb{D}}', \widetilde{\mathbb{D}}''\}$, and $\widetilde{\mathbb{Q}}$ is also split into two

random vectors as $\{\widetilde{\mathbb{Q}}', \widetilde{\mathbb{Q}}''\}$. Note here that vector $\mathbb{Q}$ functions as a splitting indicator. Namely,

if the $\#$ -th bit of $\mathbb{Q}$ is 0, $\widetilde{\mathbb{D}}'[\#]$ and $\widetilde{\mathbb{D}}''[\#]$ are set as the same as $\widetilde{\mathbb{D}}[\#]$, while $\widetilde{\mathbb{Q}}'[\#]$ and $\widetilde{\mathbb{Q}}''[\#]$ are set to two random numbers

As the MRSE is using the inner product similarity instead

of the Euclidean distance, we need to do some modifications on the data structure to fit the MRSE framework. One way to do that is by eliminating the dimension extension, the final result changes to be the inner product as $\mathbb{D} \cdot \mathbb{Q}$. However, this scheme is not good enough for our MRSE design. The major reason is that the only randomness involved is the scale factor ϕ in the trapdoor generation, which does not provide sufficient nondeterminacy in the overall scheme as required by the trapdoor

unlinkability requirement as well as the keyword privacy requirement. To provide a more advanced design for the MRSE, we now provide our MRSE_I scheme as follows.

randomness should also be carefully calibrated in the search result to obfuscate the document frequency and diminish the chances for re-identification of keywords. Introducing some randomness in the final similarity score is an effective way towards what we expect here. More specifically, unlike the randomness involved in the query vector, we insert a dummy keyword into each data vector and assign a random value to it. Each individual vector $\vec{v}_i$ is extended to $(\ell + 2)$ -dimension instead of $(\ell + 1)$, where a random variable r_i representing the dummy keyword is stored in the extended dimension. The whole scheme to achieve ranked search with multiple keywords over encrypted data is as follows

Setup The data owner randomly generates a $(\ell + 2)$ -bit vector as $\vec{v}$ and two $(\ell + 2) \times (\ell + 2)$ invertible matrices

$\{\vec{M}_1, \vec{M}_2\}$. The secret key $\mathcal{K}$ is in the form of a 3-tuple as $\{\vec{v}, \vec{M}_1, \vec{M}_2\}$.

BuildIndex($\mathcal{K}, \mathcal{D}$) The data owner generates a binary

data vector $\vec{v}_i$ for every document d_i , where each binary bit $v_i[j]$ represents whether the corresponding keyword

k_j appears in the document d_i . Subsequently, every plaintext subindex $\vec{v}_i$ is generated by applying dimension extending and splitting procedures on $\vec{v}_i$. These

procedures are similar with those in the secure kNN

computation except that the $(\ell + 1)$ -th entry in $\vec{v}_i$ is

set to a random number r_i , and the $(\ell + 2)$ -th entry in $\vec{v}_i$ is set to 1 during the dimension extending. $\vec{v}_i$

As for the efficiency, our inner product based MRSE scheme is an outstanding approach from the performance perspective. In the steps like BuildIndex or Trapdoor, the generation procedure of each subindex or trapdoor involves two multiplications of a $(\ell + 2) \times (\ell + 2)$ matrix and a $(\ell + 2)$ -dimension.

Privacy As for the *data privacy*, traditional symmetric key encryption techniques could be properly utilized here and is not within the scope of this paper. The *index privacy* is well protected if the secret key $\mathcal{K}$ is kept confidential since such vector encryption method has been proved to be secure in the known ciphertext model [28].

Table 1: Appears in every documents

Doc	Query for $\{k_1, k_2, k_3\}$	Query for $\{k_1, k_2\}$
1	$v_1 = 3, v_2 = 2(3 + v_1) + 1$	$v_1 = 2, v_2 = 1(2 + v_1) + 1$
2	$v_1 = 4, v_2 = 3(4 + v_1) + 1$	$v_1 = 1, v_2 = 1(1 + v_1) + 1$
3	$v_1 = 1, v_2 = 2(1 + v_1) + 1$	$v_1 = 0, v_2 = 1(0 + v_1) + 1$

basic scheme can generate two totally different trapdoors for the same query $\vec{q}$. This nondeterministic trapdoor generation of the deterministic property of trapdoor generation [8]. Moreover, with properly selected parameter ℓ for the random factor, even the final score results can be obfuscated very well, preventing the cloud server from learning the relationships of given trapdoors and the corresponding keywords. Note that although ℓ is expected to be small from the effectiveness point of view, the small one will introduce small obfuscation into the the final similarity scores, which may weaken the protection of keyword privacy and trapdoor unlinkability. As shown in Fig. 2, the distribution of the final similarity scores with smaller ℓ will enable the cloud server to learn more statistical information about the original similarity scores, and therefore ℓ should be set large enough from the consideration of privacy.

B. MRSE_II: Privacy-Preserving Scheme in Known Back-ground Model

When the cloud server has knowledge of some background information on the outsourced dataset, e.g., the correlation relationship of two given trapdoors, certain keyword privacy may not be guaranteed anymore by the MRSE_I scheme. This is possible in the known background model because the cloud

further combined with background information to identify the keyword in a query at high probability. After presenting how the cloud server uses scale analysis attack to break the keyword privacy, we propose a more advanced MRSE

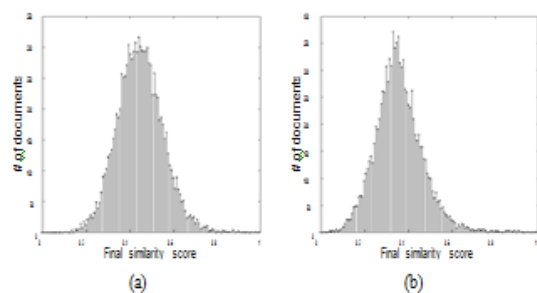


Fig. 2: Distribution of final similarity score with different standard deviations, 10k documents, 10 query keywords. (a) $\sigma = 1$. (b) $\sigma = 0.5$.

q_1 and q_2 for query keywords $\{k_1, k_2\}$ and $\{k_1, k_2, k_3\}$

respectively, there will be two special cases when searching on any three documents as listed in Tab. I and Tab. II. In any of these two cases, there exists a system of equations among final similarity scores s_i for q_1 and s'_i for q_2 as follows

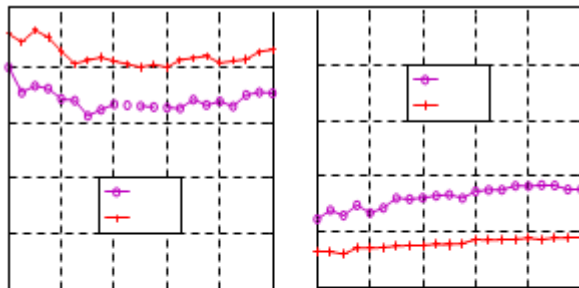
To this end, although the exact value of s_i is encrypted as s'_i , the cloud server could deduce that whether all the three documents contain k_3 or none of them contain k_3 through checking the following equivalence relationship among all

Table 2: does not appear in either document

Doc	Query for $\{k_1, k_2, k_3\}$	Query for $\{k_1, k_2\}$
1	$s_1 = 2, s'_1 = 2(2 + s_1) + 1$	$s_1 = 2, s'_1 = 2(2 + s_1) + 1$
2	$s_2 = 1, s'_2 = 2(1 + s_2) + 1$	$s_2 = 1, s'_2 = 2(1 + s_2) + 1$
3	$s_3 = 0, s'_3 = 2(0 + s_3) + 1$	$s_3 = 0, s'_3 = 2(0 + s_3) + 1$

final similarity scores in two queries,

$$\frac{s_1 - s_2}{s'_1 - s'_2} = \frac{s_2 - s_3}{s'_2 - s'_3} = \frac{s_3 - s_1}{s'_3 - s'_1}$$



$$\begin{aligned} \frac{s_1 - s_2}{s'_1 - s'_2} &= \frac{s_2 - s_3}{s'_2 - s'_3} \\ \frac{s_1 - s_2}{s'_1 - s'_2} &= \frac{s_2 - s_3}{s'_2 - s'_3} \\ \frac{s_2 - s_3}{s'_2 - s'_3} &= \frac{s_3 - s_1}{s'_3 - s'_1} \end{aligned} \quad (2)$$

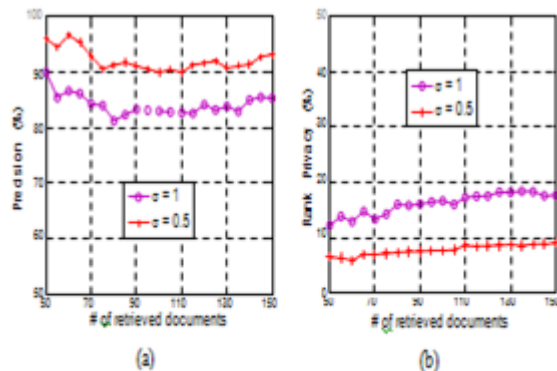
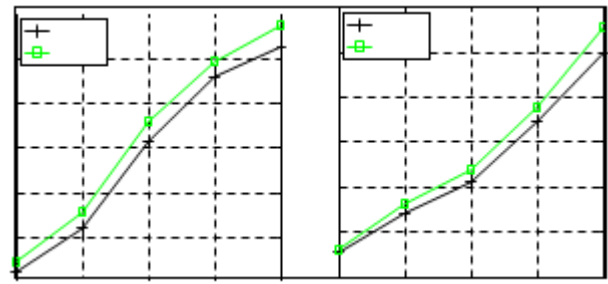


Fig. 3: With different choice of standard deviation σ for the random variable l , there exists tradeoff between (a) Precision, and (b) Rank Privacy.



V. PERFORMANCE ANALYSIS

In this section, we demonstrate a thorough experimental evaluation of the proposed technique on a real-world dataset: the Enron Email Dataset [29]. We randomly select different number of emails to build dataset. The whole experiment system is implemented by C language on a Linux Server with Intel Xeon Processor 2.93GHz. The public utility routines by Numerical Recipes are employed to compute the inverse of matrix. The performance of our technique is evaluated regarding the efficiency of two proposed MRSE schemes, as well as the tradeoff between search precision and privacy.

A. Precision and Privacy

As presented in Section IV, dummy keywords are inserted into each data vector and some of them are selected in every query. Therefore, similarity scores of documents will be not exactly accurate. In other words, when the cloud server returns top- k documents based on similarity scores of data vectors to query vector, some of real top- k relevant documents for the query may be excluded. This is because either their original similarity scores are decreased or the similarity scores of some documents out of the real top- k are increased, both of which are due to the impact of dummy keywords inserted into data vectors. To evaluate the purity of the k documents retrieved by user, we define a measure as precision $p_k = k'/k$ where k' is number of real top- k documents that are returned by the cloud server. Fig. 3(a) shows that the precision in MRSE scheme is evidently affected by the standard deviation σ of the random variable l . From the consideration of effectiveness, standard deviation σ is expected to be smaller so as to obtain high precision indicating the good purity of retrieved documents.

However, user's rank privacy may have been partially leaked to the cloud server as a consequence of small σ . As described

Frequency

1) *Index Construction:* To build a searchable subindex I_i for each document d_i in the dataset $\mathcal{D}$, the first step is to map the keyword set extracted from the document d_i to a data vector $\mathbf{v}_i$, followed by encrypting every data vector. The time cost of

mapping or encrypting depends directly on the dimensionality of data vector which is determined by the size of the dictionary, i.e., the number of indexed keywords. And the time cost of building the whole index is also related to the number of subindex which is equal to the number of documents in the dataset. Fig. 4(a) shows that, given the same dictionary where $|D| = 4000$, the time cost of building the whole index is nearly linear with the size of dataset since the time cost of building each subindex is fixed. Fig. 4(b) shows that the number of keywords indexed in the dictionary determines the time cost of building a subindex.

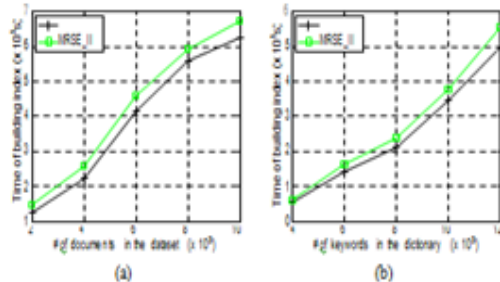


Fig. 4: Time cost of building index. (a) For the different size of dataset with the same dictionary, $n = 4000$. (b) For the same dataset with different size of dictionary, $m = 1000$.

in section III-B, the access pattern is defined as the sequence of ranked search results. Although search results cannot be protected (excluding costly PIR technique), we can still hide the rank order of retrieved documents as much as possible. In order to evaluate this privacy guarantee, we first define the rank

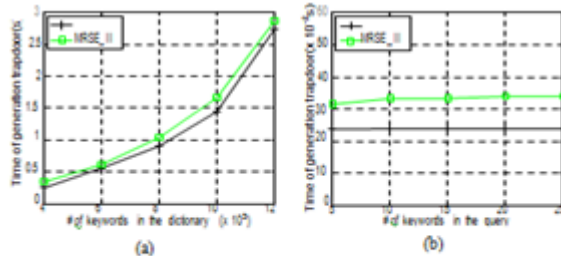


Fig. 5: Time cost of generating trapdoor. (a) For the same query keywords within different sizes of dictionary, $t = 10$. (b) For different numbers of query keywords within the same dictionary, $n = 4000$.

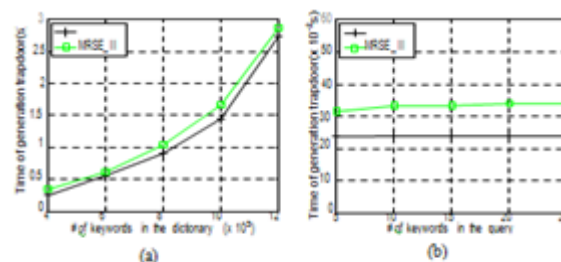


Fig. 5: Time cost of generating trapdoor. (a) For the same query keywords within different sizes of dictionary, $t = 10$. (b) For different numbers of query keywords within the same dictionary, $n = 4000$.

dimension vector where $\varnothing = |D|$, both of which have direct relationship with the size of dictionary. The

dimensionality of matrices in MRSE_II is $(\varnothing + 1) \times (\varnothing + 1)$ which is bigger than that in MRSE_I so that the index construction time becomes larger as shown in both Fig. 4(a) and Fig. 4(b). Although the time of building index is not a negligible overhead for the data owner, this is a one-time operation before data outsourcing. Besides, Tab. III lists the storage overhead of each subindex in two MRSE schemes within different sizes of dictionary. The size of subindex is absolutely linear with the dimensionality of data vector which is determined by the number of keywords in the dictionary. The sizes of subindex are very close in the two MRSE schemes because of trivial differences in the dimensionality of data vector.

2) *Trapdoor Generation*: Fig. 5(a) shows that the time to generate a trapdoor is greatly affected by the number of keywords in the dictionary. Like index construction, every trapdoor generation incurs two multiplications of a matrix and a split query vector, where the dimensionality of matrix or query vector is different in two proposed schemes and becomes larger with the increasing size of dictionary. Fig. 5(b) demonstrates the trapdoor generation cost in the MRSE_II scheme is about 20 percentages larger than that in the MRSE_I scheme. Like the subindex generation, the difference of costs to generate trapdoors is majorly caused by the different dimensionality of vector and matrices in the two MRSE schemes. More importantly, it shows that the number of query keywords has little influence on the overhead of trapdoor generation, which is a significant advantage over related works on multi-keyword searchable encryption.

3) *Query*: Query execution in the cloud server consists of computing and ranking similarity scores for all documents in the dataset. Fig. 6 shows the query time is dominated by the number of documents in the dataset while the number of keywords in the query has very slight impact on it like the cost of trapdoor generation above. With respect to the communication cost in Query, the size of the trapdoor is

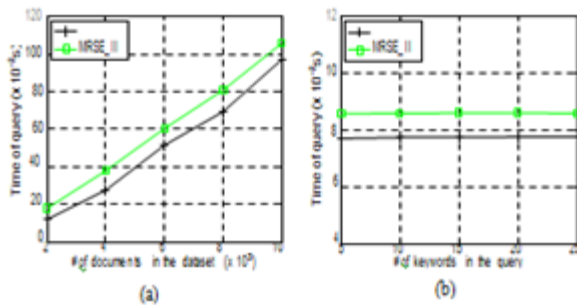


Fig. 6: Time cost of query. (a) For the same query keywords in different sizes of dataset, $t = 10$. (b) For different numbers of query keywords in the same dataset, $m = 1000$.

TABLE III: Size of subindex/trapdoor

Size of dictionary	4000	6000	8000	10000	12000
MRSE-I (KB)	31.3	46.9	62.5	78.1	93.8
MRSE-II (KB)	32.5	48.1	63.8	79.4	95.0

keeps constant given the same dictionary, no matter how many keywords are contained in a query. While the computation and communication cost in the query procedure is linear with the number of query keywords in other multiple-keyword search schemes [14], [16], our proposed schemes introduce nearly constant overhead while increasing the number of query keywords.

VI. RELATED WORK

A. Single Keyword Searchable Encryption

Traditional single keyword searchable encryption schemes [5]–[13], [22] usually build an encrypted searchable index such that its content is hidden to the server unless it is given appropriate trapdoors generated via secret key(s) [2]. It is first studied by Song et al. [5] in the symmetric key setting, and improvements and advanced security definitions are given in Goh [6], Chang et al. [7] and Curtmola et al. [8]. Our early work [22] solves secure ranked keyword search which utilizes keyword frequency to rank results instead of returning undifferentiated results. However, it only supports single keyword search. In the public key setting, Boneh et al. [9] present the first searchable encryption construction, where anyone with public key can write to the data stored on server but only authorized users with private key can search. Public key solutions are usually very computationally expensive however. Furthermore, the keyword privacy could not be protected in the public key setting since server could encrypt any keyword with public key and then use the received trapdoor to evaluate this ciphertext.

B. Boolean Keyword Searchable Encryption

To enrich search functionalities, conjunctive keyword search [14]–[18] over encrypted data have been proposed.

These schemes incur large overhead caused by their fundamental primitives, such as computation cost by bilinear map, e.g. [16], or communication cost by secret sharing, e.g. [15]. As a more general search approach, predicate encryption schemes [19]–[21] are recently proposed to support both conjunctive and disjunctive search. Conjunctive keyword search returns “all-or-nothing”, which means it only returns those documents in which all the keywords specified by the search query appear; disjunctive keyword search returns undifferentiated results, which means it returns every document that contains a subset of the specific keywords, even only one keyword of interest. In short, none of existing Boolean keyword searchable encryption schemes support multiple keywords ranked search over encrypted cloud data while preserving privacy as we propose to explore in this paper. Note that, inner product queries in predicate encryption only predicates whether two vectors are orthogonal or not, i.e., the inner product value is concealed except when it equals zero. Without providing the capability to compare concealed inner products, predicate encryption is not qualified for performing ranked search. Furthermore, most of these schemes are built upon the expensive evaluation of pairing operations on elliptic curves. Such inefficiency disadvantage also limits their practical performance when deployed in the cloud. On a different front, the research on top- k retrieval [26] in database community is also loosely connected to our problem.

VII. CONCLUSION

In this paper, for the first time we define and solve the problem of multi-keyword ranked search over encrypted cloud data, and establish a variety of privacy requirements. Among various multi-keyword semantics, we choose the efficient similarity measure of “coordinate matching”, i.e., as many matches as possible, to effectively capture the relevance of outsourced documents to the query keywords, and use “inner product similarity” to quantitatively evaluate such similarity measure. For meeting the challenge of supporting multi-keyword semantic without privacy breaches, we propose a basic idea of MRSE using secure inner product computation. Then we give two significantly improved MRSE schemes to achieve various stringent privacy requirements in two different threat models. Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given, and experiments on the real-world dataset show our proposed schemes introduce low overhead on both computation and communication.

As our future work, we will explore supporting other multi- keyword semantics (e.g., weighted query) over encrypted data, integrity check of rank order in search result and privacy guarantees in the more stronger threat model.

REFERENCES

1. L. M. Vaquero, L. Roderio-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2009.
2. S. Kamara and K. Lauter, "Cryptographic cloud storage," in *RLCPS, January 2010, LNCS. Springer, Heidelberg*.
3. Singhal, "Modern information retrieval: A brief overview," *IEEE Data Engineering Bulletin*, vol. 24, no. 4, pp. 35–43, 2001.
4. I. H. Witten, A. Moffat, and T. C. Bell, "Managing gigabytes: Compress- ing and indexing documents and images," Morgan Kaufmann Publishing, San Francisco, May 1999.
5. D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. of S&P*, 2000.
6. E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, 2003, <http://eprint.iacr.org/2003/216>.
7. Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. of ACNS*, 2005.
8. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proc. of ACM CCS*, 2006.
9. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. of EUROCRYPT*, 2004.
10. M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proc. of CRYPTO*, 2007.
11. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," *J. Cryptol.*, vol. 21, no. 3, pp. 350–391, 2008.
12. J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. of IEEE INFOCOM'10 Mini-Conference*, San Diego, CA, USA, March 2010.
13. D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. S. III, "Public key encryption that allows pir queries," in *Proc. of CRYPTO*, 2007.
14. P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. of ACNS*, 2004, pp. 31–45.
15. L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. of ICICS*, 2005.
16. D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. of TCC*, 2007, pp. 535–554.
17. R. Brinkman, "Searching in encrypted data," in *University of Twente, PhD thesis*, 2007.
18. Y. Hwang and P. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Pairing*, 2007.
19. J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. Of EUROCRYPT*, 2008.
20. A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. of EUROCRYPT*, 2010.