# Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data

Ning Cao, *Member*, *IEEE*, Cong Wang, *Member*, *IEEE*, Ming Li, *Member*, *IEEE*,
Kui Ren, *Senior Member*, *IEEE*, and Wenjing Lou, *Senior Member*, *IEEE*

**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 have 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 data in cloud computing (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. To improve search experience of the data search service, we further extend these two schemes to support more search semantics. Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given. Experiments on the real-world data set further show proposed schemes indeed introduce low overhead on computation and communication.

**Index Terms**—Cloud computing, searchable encryption, privacy-preserving, keyword search, ranked search

## 1 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 [2], [3]. 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, for example, e-mails, personal health records, photo albums, tax documents, financial transactions, and so on, may have to be encrypted by data owners before outsourcing to the commercial public cloud [4]; 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 [5]. 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

- *N. Cao is with Walmart Labs, 444 Castro St, Mountain View, CA 94041. E-mail: ncao@walmartlabs.com.*
- *C. Wang is with the Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong. E-mail: congwang@cityu.edu.hk.*
- *M. Li is with the Department of Computer Science, Utah State University, 4205 Old Main Hill, Logan, UT 84322. E-mail: ming.li@usu.edu.*
- *K. Ren is with the Department of Computer Science and Engineering, University at Buffalo, The State University of New York, 317 Davis Hall, Buffalo, NY 14260. E-mail: kuiren@buffalo.edu.*
- *W. Lou is with the Department of Computer Science, Northern Virginia Center, Virginia Polytechnic Institute and State University, 7054 Haycock Road, Falls Church, VA 22043. E-mail: wjlou@vt.edu.*

*Manuscript received 6 Dec. 2012; accepted 31 Jan. 2013; published online 15 Feb. 2013.*
*Recommended for acceptance by D. Turgut.*
*For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2012-12-1210.*
*Digital Object Identifier no. 10.1109/TPDS.2013.45.*

1045-9219/14/$31.00 © 2014 IEEE Published by the IEEE Computer Society

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” [6], 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 3.2).

In the literature, searchable encryption [7], [8], [9], [10], [11], [12], [13], [14], [15] 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, as they are developed as cryptoprimitives and cannot accommodate 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 [16], [17], [18], [19], [20], [21], [22], [23], [24] as an attempt to enrich the search flexibility, they are still not adequate to provide users with acceptable result ranking functionality (see Section 7). Our early works [25], [26] have been aware of this problem, and provide solutions 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 systemwise 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” [6], 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 ($k$NN) technique [27], 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:

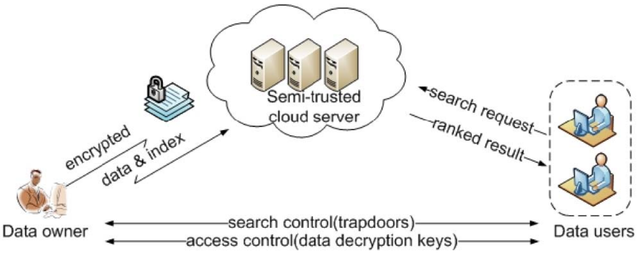

Semi-trusted cloud server
encrypted data & index
search request
ranked result
Data owner
Data users
search control(trapdoors)
access control(data decryption keys)

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

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. We investigate some further enhancements of our ranked search mechanism to support more search semantics and dynamic data operations.
4. Thorough analysis investigating privacy and efficiency guarantees of the proposed schemes is given, and experiments on the real-world data set further show the proposed schemes indeed introduce low overhead on computation and communication.

Compared with the preliminary version [1] of this paper, this journal version proposes two new mechanisms to support more search semantics. This version also studies the support of data/index dynamics in the mechanism design. Moreover, we improve the experimental works by adding the analysis and evaluation of two new schemes. In addition to these improvements, we add more analysis on secure inner product and the privacy part.

The remainder of this paper is organized as follows: In Section 2, we introduce the system model, the threat model, our design goals, and the preliminary. Section 3 describes the MRSE framework and privacy requirements, followed by Section 4, which describes the proposed schemes. Section 5 presents simulation results. We discuss related work on both single and Boolean keyword searchable encryption in Section 6, and conclude the paper in Section 7.

## 2 Problem Formulation

### 2.1 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 $\mathcal{F}$ to be outsourced to the cloud server in the encrypted form $\mathcal{C}$. To enable the searching capability over $\mathcal{C}$ for effective data utilization, the data owner, before outsourcing, will first build an encrypted searchable index $\mathcal{I}$ from $\mathcal{F}$, and then outsource both the index $\mathcal{I}$ and the encrypted document collection $\mathcal{C}$ to the cloud server. To search the document collection for $t$ given keywords, an authorized user acquires a corresponding trapdoor $T$ through search control mechanisms, for example, broadcast encryption [10]. Upon receiving $T$ 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 $k$ along with the trapdoor $T$ so that the cloud server only sends back top-$k$ documents that are most relevant to the search query. Finally, the access control mechanism [28] is employed to manage decryption capabilities given to users and the data collection can be updated in terms of inserting new documents, updating existing documents, and deleting existing documents.

### 2.2 Threat Model

The cloud server is considered as "honest-but-curious" in our model, which is consistent with related works on cloud security [28], [29]. 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.

*Known ciphertext model.* In this model, the cloud server is supposed to only know encrypted data set $\mathcal{C}$ 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 data set 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 [30] to deduce/identify certain keywords in the query.

### 2.3 Design Goals

To enable ranked search for effective utilization of outsourced 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 data set and the index, and to meet privacy requirements specified in Section 3.2.
- *Efficiency.* Above goals on functionality and privacy should be achieved with low communication and computation overhead.

### 2.4 Notations

- $\mathcal{F}$—the plaintext document collection, denoted as a set of $m$ data documents $\mathcal{F} = (F_1, F_2, \dots, F_m)$.
- $\mathcal{C}$—the encrypted document collection stored in the cloud server, denoted as $\mathcal{C} = (C_1, C_2, \dots, C_m)$.
- $\mathcal{W}$—the dictionary, i.e., the keyword set consisting of $n$ keyword, denoted as $\mathcal{W} = (W_1, W_2, \dots, W_n)$.
- $\mathcal{I}$—the searchable index associated with $\mathcal{C}$, denoted as $(I_1, I_2, \dots, I_m)$ where each subindex $I_i$ is built for $F_i$.
- $\widetilde{\mathcal{W}}$—the subset of $\mathcal{W}$, representing the keywords in a search request, denoted as $\widetilde{\mathcal{W}} = (W_{j_1}, W_{j_2}, \dots, W_{j_t})$.
- $T_{\widetilde{\mathcal{W}}}$—the trapdoor for the search request $\widetilde{\mathcal{W}}$.
- $\mathcal{F}_{\widetilde{\mathcal{W}}}$—the ranked id list of all documents according to their relevance to $\widetilde{\mathcal{W}}$.

### 2.5 Preliminary on Coordinate Matching

As a hybrid of conjunctive search and disjunctive search, "coordinate matching" [6] 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 data set 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 specify a list of keywords indicating their interest and retrieve the most relevant documents with a rank order.

## 3 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 systemwise privacy requirements for such a secure cloud data utilization system.

### 3.1 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 $(1^\ell)$. *Taking a security parameter $\ell$ as input, the data owner outputs a symmetric key as $SK$.*
- BuildIndex $(\mathcal{F}, SK)$. *Based on the data set $\mathcal{F}$, the data owner builds a searchable index $\mathcal{I}$ which is encrypted by the symmetric key $SK$ and then outsourced to the cloud server. After the index construction, the document collection can be independently encrypted and outsourced.*
- Trapdoor $(\widetilde{\mathcal{W}})$. *With t keywords of interest in $\widetilde{\mathcal{W}}$ as input, this algorithm generates a corresponding trapdoor $T_{\widetilde{\mathcal{W}}}$.*
- Query $(T_{\widetilde{\mathcal{W}}}, k, \mathcal{I})$. *When the cloud server receives a query request as $(T_{\widetilde{\mathcal{W}}}, k)$, it performs the ranked search on the index $\mathcal{I}$ with the help of trapdoor $T_{\widetilde{\mathcal{W}}}$, and finally returns $\mathcal{F}_{\widetilde{\mathcal{W}}}$, the ranked id list of top-k documents sorted by their similarity with $\widetilde{\mathcal{W}}$.*

Neither the search control nor the access control is within the scope of this paper. While the former is to regulate how authorized users acquire trapdoors, the later is to manage users' access to outsourced documents.

### 3.2 Privacy Requirements for MRSE

The representative privacy guarantee in the related literature, such as searchable encryption, is that the server should learn nothing but search results. With this general privacy description, we explore and establish a set of strict privacy requirements specifically for the MRSE framework.

As for the *data privacy*, the data owner can resort to the traditional symmetric key cryptography to encrypt the data before outsourcing, and successfully prevent the cloud server from prying into the outsourced data. With respect to the *index privacy*, if the cloud server deduces any association between keywords and encrypted documents from index, it may learn the major subject of a document, even the content of a short document [30]. Therefore, the searchable index should be constructed to prevent the cloud server from performing such kind of association attack. While data and index privacy guarantees are demanded by default in the related literature, various *search privacy* requirements involved in the query procedure are more complex and difficult to tackle as follows.

*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 [31]. When the cloud server knows some background information of the data set, 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, for example, 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 trapdoor generation procedure.

*Access pattern.* Within the ranked search, the access pattern is the sequence of search results where every search result is a set of documents with rank order. Specifically, the search result for the query keyword set $\widetilde{\mathcal{W}}$ is denoted as $\mathcal{F}_{\widetilde{\mathcal{W}}}$, consisting of the id list of all documents ranked by their relevance to $\widetilde{\mathcal{W}}$. Then the access pattern is denoted as $(\mathcal{F}_{\widetilde{\mathcal{W}}_1}, \mathcal{F}_{\widetilde{\mathcal{W}}_2}, \ldots)$ which are the results of sequential searches. Although a few searchable encryption works, for example, [19] has been proposed to utilize private information retrieval (PIR) technique [32], to hide the access pattern, our proposed schemes are not designed to protect the access pattern for the efficiency concerns. This is because any PIR-based technique must "touch" the whole data set outsourced on the server which is inefficient in the large-scale cloud system.

## 4 Privacy-Preserving and Efficient MRSE

To efficiently achieve multi-keyword ranked search, we propose to employ "inner product similarity" [6] to quantitatively evaluate the efficient similarity measure "coordinate matching." Specifically, $D_i$ is a binary data vector for document $F_i$ where each bit $D_i[j] \in \{0, 1\}$ represents the existence of the corresponding keyword $W_j$ in that document, and $Q$ is a binary query vector indicating the keywords of interest where each bit $Q[j] \in \{0, 1\}$ represents the existence of the corresponding keyword $W_j$ in the query $\widetilde{\mathcal{W}}$. The similarity score of document $F_i$ to query $\widetilde{\mathcal{W}}$ is therefore expressed as the inner product of their binary column vectors, i.e., $D_i \cdot Q$. 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 systemwise privacy, data vector $D_i$, query vector $Q$ and their inner product $D_i \cdot Q$ 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 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. We further discuss supporting more search semantics and dynamic operation.

### 4.1 Secure Inner Product Computation

In the secure kNN scheme [27], euclidean distance between a data record $p_i$ and a query vector $q$ is used to select $k$ nearest database records. The secret key is composed of one $(d+1)$-bit vector as $S$ and two $(d+1) \times (d+1)$ invertible matrices as $\{M_1, M_2\}$, where $d$ is the number of fields for each record $p_i$. First, every data vector $p_i$ and query vector $q$ are extended to $(d+1)$-dimension vectors as $\vec{p_i}$ and $\vec{q}$, where the $(d+1)$th dimension is set to $-0.5\|p_i^2\|$ and 1, respectively. Besides, the query vector $\vec{q}$ is scaled by a random number $r > 0$ as $(rq, r)$. Then, $\vec{p_i}$ is split into two random vectors as $\{\vec{p_i}', \vec{p_i}''\}$, and $\vec{q}$ is also split into two random vectors as $\{\vec{q}\,', \vec{q}\,''\}$. Note here that vector $S$ functions as a splitting indicator. Namely, if the $j$th bit of $S$ is 0, $\vec{p_i}'[j]$ and $\vec{p_i}''[j]$ are set as the same as $\vec{p_i}[j]$, while $\vec{q}\,'[j]$ and $\vec{q}\,''[j]$ are set to two random numbers so that their sum is equal to $\vec{q}[j]$; if the $j$th bit of $S$ is 1, the splitting process is similar except that $\vec{p_i}$ and $\vec{q}$ are switched. The split data vector pair $\{\vec{p_i}', \vec{p_i}''\}$ is encrypted as $\{M_1^T \vec{p_i}', M_2^T \vec{p_i}''\}$, and the split query vector pair $\{\vec{q}\,', \vec{q}\,''\}$ is encrypted as $\{M_1^{-1}\vec{q}\,', M_2^{-1}\vec{q}\,''\}$. In the query step, the product of data vector pair and query vector pair, i.e., $-0.5r(\|p_i\|^2 - 2p_i \cdot q)$, is serving as the indicator of euclidean distance $(\|p_i\|^2 - 2p_i \cdot q + \|q\|^2)$ to select $k$ nearest neighbors.

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 $rp_i \cdot q$. While the encryption of either data record or query vector involves two multiplications of a $d \times d$ matrix and a $d$-dimension vector with complexity $O(d^2)$, the final inner product computation involves two multiplications of two $d$-dimension vectors with complexity $O(d)$. In the known ciphertext model, the splitting vector $S$ is unknown, so $\vec{p_i}'$ and $\vec{p_i}''$ are considered as two random $d$-dimensional vectors. To solve the linear equations created by the encryption of data vectors, we have $2dm$ unknowns in $m$ data vectors and $2d^2$ unknowns in $\{M_1, M_2\}$. Since we have only $2dm$ equations, which are less than the number of unknowns, there is no sufficient information to solve either data vectors or $\{M_1, M_2\}$. Similarly, $\vec{q}\,'$ and $\vec{q}\,''$ are also considered as two random $d$-dimensional vectors. To solve the linear equations created by the encryption of query vectors, we have $2d$ unknowns in two query vectors and $2d^2$ unknowns in $\{M_1, M_2\}$. Since we have only $2d$ equations here, which are less than the number of unknowns, there is no sufficient information to solve either query vectors or $\{M_1, M_2\}$. Hence, we believe that without prior knowledge of secret key, neither data vector nor query vector, after such a series of processes like splitting and multiplication, can be recovered by analyzing their corresponding ciphertexts.

## 4.2 MRSE_I: Privacy-Preserving Scheme in Known Ciphertext Model

The adapted secure inner product computation scheme is not good enough for our MRSE design. The major reason is that the only randomness involved is the scale factor $r$ 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.

### 4.2.1 MRSE_I Scheme

In our more advanced design, instead of simply removing the extended dimension in the query vector as we plan to do at the first glance, we preserve this dimension extending operation but assign a new random number $t$ to the extended dimension in each query vector. Such a newly added randomness is expected to increase the difficulty for the cloud server to learn the relationship among the received trapdoors. In addition, as mentioned in the keyword privacy requirement, randomness should also be carefully calibrated in the search result to obfuscate the document frequency and diminish the chances for reidentification of keywords. Introducing some randomness in the final similarity score is an effective way toward 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 $D_i$ is extended to $(n+2)$-dimension instead of $(n+1)$, where a random variable $\varepsilon_i$ 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 $(n+2)$-bit vector as $S$ and two $(n+2)\times(n+2)$ invertible matrices $\{M_1, M_2\}$. The secret key $SK$ is in the form of a 3-tuple as $\{S, M_1, M_2\}$.
- BuildIndex $(\mathcal{F}, SK)$. The data owner generates a binary data vector $D_i$ for every document $F_i$, where each binary bit $D_i[j]$ represents whether the corresponding keyword $W_j$ appears in the document $F_i$. Subsequently, every plaintext subindex $\vec{D}_i$ is generated by applying dimension extending and splitting procedures on $D_i$. These procedures are similar with those in the secure kNN computation except that the $(n+1)$th entry in $\vec{D}_i$ is set to a random number $\varepsilon_i$, and the $(n+2)$th entry in $\vec{D}_i$ is set to 1 during the dimension extending. $\vec{D}_i$ is therefore equal to $(D_i, \varepsilon_i, 1)$. Finally, the subindex $I_i = \{M_1^T \vec{D}_i{}', M_2^T \vec{D}_i{}''\}$ is built for every encrypted document $C_i$.
- Trapdoor $(\widetilde{\mathcal{W}})$. With $t$ keywords of interest in $\widetilde{\mathcal{W}}$ as input, one binary vector $Q$ is generated where each bit $Q[j]$ indicates whether $W_j \in \widetilde{\mathcal{W}}$ is true or false. $Q$ is first extended to $n+1$-dimension which is set to 1, and then scaled by a random number $r \neq 0$, and finally extended to a $(n+2)$-dimension vector as $\vec{Q}$ where the last dimension is set to another random number $t$. $\vec{Q}$ is therefore equal to $(rQ, r, t)$. After applying the same splitting and encrypting processes as above, the trapdoor $T_{\widetilde{\mathcal{W}}}$ is generated as $\{M_1^{-1}\vec{Q}', M_2^{-1}\vec{Q}''\}$.
- Query $(T_{\widetilde{\mathcal{W}}}, k, \mathcal{I})$. With the trapdoor $T_{\widetilde{\mathcal{W}}}$, the cloud server computes the similarity scores of each document $F_i$ as in (1). WLOG, we assume $r > 0$. After sorting all scores, the cloud server returns the top-$k$ ranked id list $\mathcal{F}_{\widetilde{\mathcal{W}}}$.

With $t$ brought into the query vector and $\varepsilon_i$ brought into each data vector, the final similarity scores would be

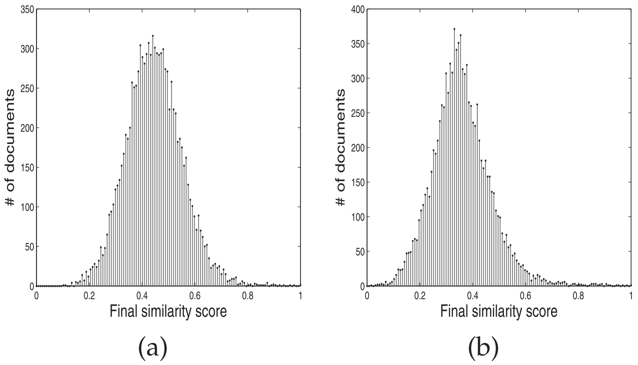
$$\begin{aligned} I_i \cdot T_{\widetilde{\mathcal{W}}} &= \{M_1^T \vec{D}_i{}', M_2^T \vec{D}_i{}''\} \cdot \{M_1^{-1}\vec{Q}', M_2^{-1}\vec{Q}''\} \\ &= \vec{D}_i{}' \cdot \vec{Q}' + \vec{D}_i{}'' \cdot \vec{Q}'' \\ &= \vec{D}_i \cdot \vec{Q} \\ &= (D_i, \varepsilon_i, 1) \cdot (rQ, r, t) \\ &= r(D_i \cdot Q + \varepsilon_i) + t. \end{aligned} \tag{1}$$

Note that in the original case, the final score is simply $rD_i \cdot q$, which preserves the scale relationship for two queries on the same keywords. But such an issue is no longer valid in our improved scheme due to the randomness of both $t$ and $\varepsilon_i$, which clearly demonstrates the effectiveness and improved security strength of our MSRE_I mechanism.

### 4.2.2 Analysis

We analyze this MRSE_I scheme from three aspects of design goals described in Section 2.

*Functionality and efficiency.* Assume the number of query keywords appearing in a document $F_i$ is $x_i = D_i \cdot Q$. From (1), the final similarity score as $y_i = I_i \cdot T_{\widetilde{\mathcal{W}}} = r(x_i + \varepsilon_i) + t$ is a linear function of $x_i$, where the coefficient $r$ is set as a positive random number. However, because the random factor $\varepsilon_i$ is introduced as a part of the similarity score, the final search result on the basis of sorting similarity scores may not be as accurate as that in original scheme. For the consideration of search accuracy, we can let $\varepsilon_i$ follow a normal distribution $N(\mu, \sigma^2)$, where the standard deviation $\sigma$ functions as a flexible tradeoff parameter among search accuracy and security. From the consideration of effectiveness, $\sigma$ is expected to be smaller so as to obtain high precision indicating the good purity of retrieved



(a) (b)

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

documents. To quantitatively evaluate the search accuracy, we set a measure as precision $P_k$ to capture the fraction of returned top-$k$ documents that are included in the real top-$k$ list. Detailed accuracy evaluation on the real-world data set will be given in Section 5.

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 $(n+2) \times (n+2)$ matrix and a $(n+2)$-dimension vector. In the Query, the final similarity score is computed through two multiplications of two $(n+2)$-dimension vectors.

*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 $SK$ is kept confidential since such vector encryption method has been proved to be secure in the known ciphertext model [27]. Although we add two more dimensions to the vectors compared to the adapted secure inner product computation, the number of equations as $2(n+2)m$ is still less than the number of unknowns as the sum of $2(n+2)m$ unknowns in $m$ data vectors and $2d^2$ unknowns in $\{M_1, M_2\}$. With the randomness introduced by the splitting process and the random numbers $r$, and $t$, our basic scheme can generate two totally different trapdoors for the same query $\widetilde{W}$. This nondeterministic trapdoor generation can guarantee the *trapdoor unlinkability* which is an unsolved privacy leakage problem in related symmetric key-based searchable encryption schemes because of the deterministic property of trapdoor generation [10]. Moreover, with properly selected parameter $\sigma$ for the random factor $\varepsilon_i$, 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 $\sigma$ is expected to be small from the effectiveness point of view, the small one will introduce small obfuscation into 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 $\sigma$ will enable the cloud server to learn more statistical information about the original similarity scores, and therefore $\sigma$ should be set large enough from the consideration of privacy.

TABLE 1
$K_3$ Appears in Every Document

| Doc | Query for $\{K_1, K_2, K_3\}$ | Query for $\{K_1, K_2\}$ |
|---|---|---|
| 1 | $x_1 = 3, y_1 = r(3 + \varepsilon_1) + t$ | $x'_1 = 2, y'_1 = r'(2 + \varepsilon_1) + t'$ |
| 2 | $x_2 = 2, y_2 = r(2 + \varepsilon_2) + t$ | $x'_2 = 1, y'_2 = r'(1 + \varepsilon_2) + t'$ |
| 3 | $x_3 = 1, y_3 = r(1 + \varepsilon_3) + t$ | $x'_3 = 0, y'_3 = r'(0 + \varepsilon_3) + t'$ |

### 4.3 MRSE_II: Privacy-Preserving Scheme in Known Background Model

When the cloud server has knowledge of some background information on the outsourced data set, for example, 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 server can use scale analysis as follows to deduce the keyword specific information, for example, document frequency, which can be 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 scheme to be privacy-preserving in the known background model.

#### *4.3.1 Scale Analysis Attack*

Given two correlated trapdoors $T_1$ and $T_2$ 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 Tables 1 and 2. In any of these two cases, there exists a system of equations among final similarity scores $y_i$ for $T_1$ and $y'_i$ for $T_2$ as follows:

$$\begin{cases} y_1 - y_2 = r(1 + \varepsilon_1 - \varepsilon_2); \\ y'_1 - y'_2 = r'(1 + \varepsilon_1 - \varepsilon_2); \\ y_2 - y_3 = r(1 + \varepsilon_2 - \varepsilon_3); \\ y'_2 - y'_3 = r'(1 + \varepsilon_2 - \varepsilon_3); \\ y_1 - y_3 = r(2 + \varepsilon_1 - \varepsilon_3); \\ y'_1 - y'_3 = r'(2 + \varepsilon_1 - \varepsilon_3). \end{cases} \tag{2}$$

To this end, although the exact value of $x_i$ is encrypted as $y_i$, 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 final similarity scores in two queries

$$\frac{y_1 - y_2}{y'_1 - y'_2} = \frac{y_2 - y_3}{y'_2 - y'_3} = \frac{y_1 - y_3}{y'_1 - y'_3}. \tag{3}$$

By extending three documents to the whole data set, the cloud server could further deduce two possible values of document frequency of keyword $K_3$. In the known background model, the server can identify the keyword $K_3$ by

TABLE 2
$K_3$ Does Not Appear in Either Document

| Doc | Query for $\{K_1, K_2, K_3\}$ | Query for $\{K_1, K_2\}$ |
|---|---|---|
| 1 | $x_1 = 2, y_1 = r(2 + \varepsilon_1) + t$ | $x'_1 = 2, y'_1 = r'(2 + \varepsilon_1) + t'$ |
| 2 | $x_2 = 1, y_2 = r(1 + \varepsilon_2) + t$ | $x'_2 = 1, y'_2 = r'(1 + \varepsilon_2) + t'$ |
| 3 | $x_3 = 0, y_3 = r(0 + \varepsilon_3) + t$ | $x'_3 = 0, y'_3 = r'(0 + \varepsilon_3) + t'$ |

referring to the keyword specific document frequency information about the data set.

#### 4.3.2 MRSE_II Scheme

The privacy leakage shown above is caused by the fixed value of random variable $\varepsilon_i$ in data vector $D_i$. To eliminate such fixed property in any specific document, more dummy keywords instead of only one should be inserted into every data vector $D_i$. All the vectors are extended to $(n+U+1)$-dimension instead of $(n+2)$, where $U$ is the number of dummy keywords inserted. Improved details in the MRSE_II scheme is presented as follows:

- Setup $(1^n)$. The data owner randomly generates a $(n+U+1)$-bit vector as $S$ and two $(n+U+1) \times (n+U+1)$ invertible matrices $\{M_1, M_2\}$.
- BuildIndex $(\mathcal{F}, SK)$. The $(n+j+1)$th entry in $\vec{D}_i$ where $j \in [1, U]$ is set to a random number $\varepsilon^{(j)}$ during the dimension extending.
- Trapdoor $(\widetilde{\mathcal{W}})$. By randomly selecting $V$ out of $U$ dummy keywords, the corresponding entries in $Q$ are set to 1.
- Query $(T_{\widetilde{\mathcal{W}}}, k, \mathcal{I})$. The final similarity score computed by cloud server is equal to $r(x_i + \sum \varepsilon_i^{(v)}) + t_i$ where the $v$th dummy keyword is included in the $V$ selected ones.

#### 4.3.3 Analysis

Assume the probability of two $\sum \varepsilon_i^{(v)}$ having the same value should be less than $1/2^\omega$, it then means there should be at least $2^\omega$ different values of $\sum \varepsilon_i^{(v)}$ for each data vector. The number of different $\sum \varepsilon_i^{(v)}$ is not larger than $\binom{U}{V}$, which is maximized when $\frac{U}{V} = 2$. Besides, considering $\binom{U}{V} \geq (\frac{U}{V})^V = 2^V$, it is greater than $2^\omega$ when $U = 2\omega$ and $V = \omega$. So every data vector should include at least $2\omega$ dummy entries, and every query vector will randomly select half dummy entries. Here, $\omega$ can be considered as a system parameter for the tradeoff between efficiency and privacy. With properly setting the value of $\omega$, the MRSE_II scheme is secure against scale analysis attack, and provides various expected privacy guarantees within the known ciphertext model or the known background model.

Moreover, every $\varepsilon^{(j)}$ is assumed to follow the same uniform distribution $M(\mu' - c, \mu' + c)$, where the mean is $\mu'$ and the variance as $\sigma'^2$ is $c^2/3$. According to the central limit theorem, the sum of $\omega$ independent random variables $\varepsilon^{(j)}$ follows the Normal distribution, where the mean is $\omega\mu'$ and the variance is $\omega\sigma'^2 = \omega c^2/3$. To make $\sum \varepsilon_i^{(v)}$ follow the Normal distribution $N(\mu, \sigma^2)$ as above, the value of $\mu'$ is set as $\mu/\omega$ and the value of $c$ is set as $\sqrt{\frac{3}{\omega}}\sigma$ so that $\omega\mu' = \mu$ and $\omega\sigma'^2 = \sigma^2$. With such parameter setting, search accuracy is statistically the same as that in MRSE_I scheme.

### 4.4 MRSE_I_TF

In the ranking principle "coordinate matching," the presence of keyword in the document or the query is shown as 1 in the data vector or the query vector. Actually, there are more factors which could make impact on the search usability. For example, when one keyword appears in most documents in the data set, the importance of this keyword in the query is less than other keywords which appears in less documents. Similarly, if one document contains a query keyword in multiple locations, the user may prefer this to the other document which contains the query keyword in only one location. To capture these information in the search process, we use the TF × IDF weighting rule within the vector space model to calculate the similarity, where TF (or term frequency) is the number of times a given term or keyword (we will use them interchangeably hereafter) appears within a file (to measure the importance of the term within the particular file), and IDF (or inverse document frequency) is obtained by dividing the number of files in the whole collection by the number of files containing the term (to measure the overall importance of the term within the whole collection). Among several hundred variations of the TF × IDF weighting scheme, no single combination of them outperforms any of the others universally [33]. Thus, without loss of generality, we choose an example formula that is commonly used and widely seen in the literature (see [5, chapter 4]) for the relevance score calculation

$$Score(F_i, Q) = \frac{1}{|F_i|} \sum_{W_j \in \widetilde{\mathcal{W}}} (1 + \ln f_{i,j}) \cdot \ln\left(1 + \frac{m}{f_j}\right). \quad (4)$$

Here $f_{i,j}$ denotes the TF of keyword $W_j$ in file $F_i$; $f_j$ denotes the number of files that contain keyword $W_j$ which is called document frequency; $m$ denotes the total number of files in the collection; and $|F_i|$ is the euclidean length of file $F_i$, obtained by

$$\sqrt{\sum_{j=1}^{n} (1 + \ln f_{i,j})^2},$$

functioning as the normalization factor.

To calculate the relevance score as shown in (4) on the server side, we propose a new search mechanism MRSE_I_TF as follows which modify related data structures in the previous scheme MRSE_I. As for the dictionary $\mathcal{W}$, the document frequency $f_j$ is attached to every keyword $W_j$, which will be used in the generation of query vector. In BuildIndex, for every keyword $W_j$ appearing in the document $F_i$, the corresponding entry $D_i[j]$ in the data vector $D_i$ is changed from a binary value 1 to the normalized term frequency, i.e., $\frac{1+\ln f_{i,j}}{|F_i|}$. Similarly, the query vector $Q$ changes corresponding entries from 1 to $\ln(1 + \frac{m}{f_j})$. Finally, the similarity score is as follows:

$$\begin{aligned} I_i \cdot T_{\widetilde{\mathcal{W}}} &= r(D_i \cdot Q + \varepsilon_i) + t \\ &= r\left(\sum_{W_j \in Q} \frac{1 + \ln f_{i,j}}{|F_i|} \cdot \ln\left(1 + \frac{m}{f_j}\right) + \varepsilon_i\right) + t \quad (5) \\ &= r(Score(F_i, Q) + \varepsilon_i) + t. \end{aligned}$$

Therefore, the similarity of the document and the query in terms of the cosine of the angle between the document vector and the query vector could be evaluated by computing the inner product of subindex $I_i$ and trapdoor $T_{\widetilde{\mathcal{W}}}$. Although this similarity measurement introduces more computation cost during the index construction and trapdoor generation, it captures more related information on the content of documents and query which returns better results of users' interest. As we will see in Section 5, the additional cost of this measurement in BuildIndex and

Trapdoor is relatively small compared to the whole cost. Besides, BuildIndex is a one-time computation for the whole scheme.

### 4.5 MRSE_II_TF

Here, although some entries in $D_i$ have been changed from binary value 1 to normalized term frequency, the scale analysis attack presented in Section 4.3 still partially works in the known background model. With similar setting in the previous section, the first query contains two keywords as $\{K_1, K_2\}$ while the second query contains three keywords as $\{K_1, K_2, K_3\}$. Given three documents as an example, the first keyword $K_1$ appears in two documents as $F_1$ and $F_2$, and the second keyword $K_2$ appears in document $F_1$. Note that there are some differences between this attack and previous one. If the third keyword $K_3$ appears in each of these three documents as shown in Table 1, such equivalence relationship as shown in (3) does not exist among these documents here. Here we only consider the case that the third keyword $K_3$ does not appear in any of these three documents. The final similarity scores are shown in (6).

Recall that the scale analysis attack presented in Section 4.3, it is caused by the fixed value of random variable $\varepsilon_i$ in each data vector $D_i$ which remains same here. From (6), the cloud server can still deduce the equivalence relationship as presented in (3). As a result, the document frequency could be exposed to cloud server and further used to identify this keyword in the known background model. To this end, we can employ the same solution as presented in MRSE_II to build the new mechanism as MRSE_II_TF where more dummy keywords instead of only one are inserted into data vectors

$$\begin{cases} y_1 = r\Big(\dfrac{1+\ln f_{1,1}}{|F_1|}\cdot \ln\left(1+\dfrac{m}{f_1}\right) \\ \qquad +\dfrac{1+\ln f_{1,2}}{|F_1|}\cdot \ln\left(1+\dfrac{m}{f_2}\right)+\varepsilon_1\Big)+t; \\ y_2 = r\Big(\dfrac{1+\ln f_{2,1}}{|F_2|}\cdot \ln\left(1+\dfrac{m}{f_1}\right)+\varepsilon_2\Big)+t; \\ y_3 = r\varepsilon_3+t; \\ y'_1 = r'\Big(\dfrac{1+\ln f_{1,1}}{|F_1|}\cdot \ln\left(1+\dfrac{m}{f_1}\right) \\ \qquad +\dfrac{1+\ln f_{1,2}}{|F_1|}\cdot \ln\left(1+\dfrac{m}{f_2}\right)+\varepsilon_1\Big)+t'; \\ y'_2 = r'\Big(\dfrac{1+\ln f_{2,1}}{|F_2|}\cdot \ln\left(1+\dfrac{m}{f_1}\right)+\varepsilon_2\Big)+t'; \\ y'_3 = r'\varepsilon_3+t'. \end{cases} \tag{6}$$

### 4.6 Supporting Data Dynamics

After the data set is outsourced to the cloud server, it may be updated in addition to being retrieved [34]. Along with the updating operation on data documents, supporting the score dynamics in the searchable index is thus of practical importance. While we consider three dynamic data operations as inserting new documents, modifying existing documents, and deleting existing documents, corresponding operations on the searchable index includes generating new index, updating existing index, and deleting existing index. Since dynamic data operations also affect the document frequency of corresponding keywords, we also need to update the dictionary $\mathcal{W}$.

For the operation of inserting new documents in the data set, there may be some new keywords in new documents which need to be inserted in the dictionary $\mathcal{W}$. Remember that every subindex in our scheme has fixed dimension as same as the number of keywords in the old dictionary, so the straightforward solution is to retrieve all the subindexes from the cloud server, and then decrypt, rebuild, and encrypt them before outsourcing to the cloud server. However, this approach introduces much cost on computation and communication for both sides which is impractical in the "pay-as-you-use" cloud paradigm. To reduce such great cost, we preserve some blank entries in the dictionary and set corresponding entries in each data vector as 0. If the dictionary needs to index new keywords in the case of inserting new documents, we just replace the blank entries in the dictionary by new keywords, and generate subindexes for new documents based on the updated dictionary. The other documents and their subindexes stored on the cloud server are not affected and therefore remain the same as before. The number of preserved entries functions as a tradeoff parameter to balance the storage cost and the system scalability.

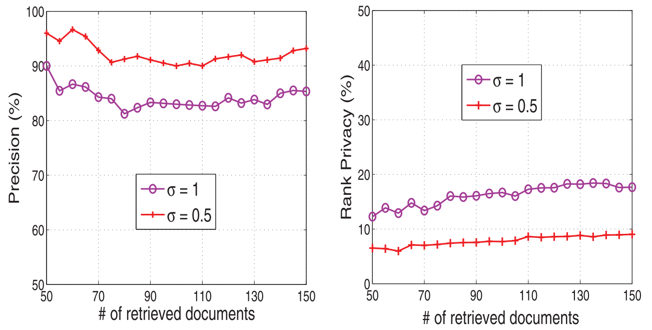
When existing documents are modified, corresponding subindexes are also retrieved from the cloud server and then updated in terms of the term frequency before outsourcing. If new keywords are introduced during the modification operation, we utilize the same method which is proposed in the previous insertion operation. As a special case of modification, the operation of deleting existing documents introduce less computation and communication cost since it only requires to update the document frequency of all the keywords contained by these documents.

## 5 Performance Analysis

In this section, we demonstrate a thorough experimental evaluation of the proposed technique on a real-world data set: the Enron Email Data Set [35]. We randomly select different number of e-mails to build data set. The whole experiment system is implemented by C language on a Linux Server with Intel Xeon Processor 2.93 GHz. 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 four proposed MRSE schemes, as well as the tradeoff between search precision and privacy.

### 5.1 Precision and Privacy

As presented in Section 4, 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.



(a) (b)

Fig. 3. With different choice of standard deviation $\sigma$ for the random variable $\varepsilon$, there exists tradeoff between (a) Precision, and (b) Rank Privacy.

Fig. 3a shows that the precision in MRSE scheme is evidently affected by the standard deviation $\sigma$ of the random variable $\varepsilon$. From the consideration of effectiveness, standard deviation $\sigma$ is expected to be smaller so as to obtain high precision indicating the good purity of retrieved documents.

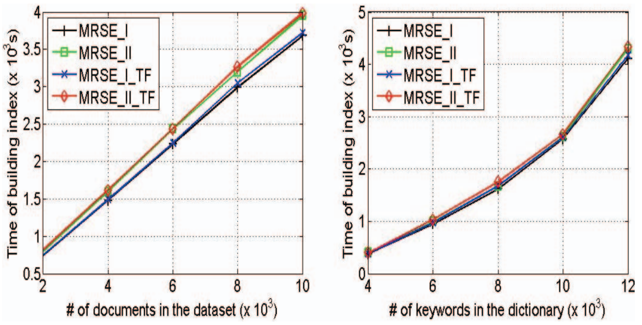
However, user's rank privacy may have been partially leaked to the cloud server as a consequence of small $\sigma$. As described in Section 3.2, the access pattern is defined as the sequence of ranked search results. Although search results cannot be protected (excluding costly PIR technique), we can still hide the rank order of retrieved documents as much as possible. To evaluate this privacy guarantee, we first define the rank perturbation as $\widetilde{p}_i = |r_i - r'_i|/k$, where $r_i$ is the rank number of document $F_i$ in the retrieved top-$k$ documents and $r'_i$ is its rank number in the real ranked documents. The overall rank privacy measure at point $k$ is then defined as the average of all the $\widetilde{p}_i$ for every document $i$ in the retrieved top-$k$ documents, denoted as $\widetilde{P}_k = \sum \widetilde{p}_i/k$. Fig. 3b shows the rank privacy at different points with two standard deviations $\sigma = 1$ and $\sigma = 0.5$, respectively.

From these two figures, we can see that small $\sigma$ leads to higher precision of search result but lower rank privacy guarantee, while large $\sigma$ results in higher rank privacy guarantee but lower precision. In other words, our scheme provides a balance parameter for data users to satisfy their different requirements on precision and rank privacy.

## 5.2 Efficiency

### 5.2.1 Index Construction

To build a searchable subindex $I_i$ for each document $F_i$ in the data set $\mathcal{F}$, the first step is to map the keyword set extracted from the document $F_i$ to a data vector $D_i$, 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 data set. Fig. 4a shows that, given the same dictionary where $|\mathcal{W}| = 4{,}000$, the time cost of building the whole index is nearly linear with the size of data set since the time cost of building each subindex is



(a) (b)

Fig. 4. Time cost of building index. (a) For the different size of data set with the same dictionary, $n = 4{,}000$. (b) For the same data set with different size of dictionary, $m = 1{,}000$.

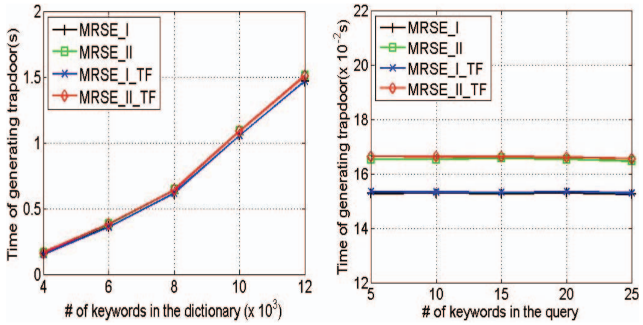
fixed. Fig. 4b shows that the number of keywords indexed in the dictionary determines the time cost of building a subindex. As presented in the Section 4.2, the major computation to generate a subindex in MRSE_I includes the splitting process and two multiplications of a $(n+2) \times (n+2)$ matrix and a $(n+2)$-dimension vector where $n = |\mathcal{W}|$, both of which have direct relationship with the size of dictionary. The dimensionality of matrices in MRSE_II is $(n+U+1) \times (n+U+1)$ so that its index construction time with complexity $O(m(n+U)^2)$ is bigger than that in MRSE_I with complexity $O(mn^2)$ as shown in Figs. 4a and 4b. Both the MRSE_I_TF and the MRSE_II_TF, presented in Sections 4.4 and 4.5, respectively, introduce more computation during the index construction since we need to collect the term frequency information for each keyword in every document and then perform the normalization calculation. But, as shown in both figures, such additional computation in the TF × IDF weighting rule is insignificant considering much more computation are caused by the splitting process and matrix multiplication. 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, Table 3 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.

### 5.2.2 Trapdoor Generation

Fig. 5a 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

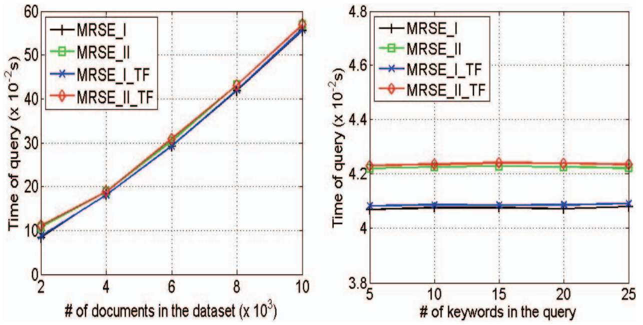
TABLE 3
Size of Subindex/Trapdoor

| 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 |



(a) (b)

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, $n = 4{,}000$.



(a) (b)

Fig. 6. Time cost of query. (a) For the same query keywords in different sizes of data set, $t = 10$. (b) For different numbers of query keywords in the same data set, $m = 1{,}000$.

in two proposed schemes and becomes larger with the increasing size of dictionary. Fig. 5b demonstrates the trapdoor generation cost in the MRSE_II scheme with complexity $O((n+U)^2)$ is about 10 percent larger than that in the MRSE_I scheme with complexity $O(n^2)$. The MRSE_I_TF and MRSE_II_TF have similar difference where the additional logarithm computation accounts for very small proportion of the whole trapdoor generation. Like the subindex generation, the difference of costs to generate trapdoors is mainly 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.

### 5.2.3 Query

Query execution in the cloud server consists of computing and ranking similarity scores for all documents in the data set. The computation of similarity scores for the whole data collection is $O(mn)$ in MRSE_I and MRSE_I_TF, and the computation increases to $O(m(n+U))$ in MRSE_II and MRSE_II_TF. Fig. 6 shows the query time is dominated by the number of documents in the data set while the number of keywords in the query has very slight impact on it like the cost of trapdoor generation above. The two schemes in the known ciphertext model as MRSE_I and MRSE_I_TF have very similar query speed since they have the same dimensionality which is the major factor deciding the computation cost in the query. The query speed difference between MRSE_I and MRSE_I_TF or between MRSE_II and MRSE_II_TF is also caused by the dimensionality of data vector and query vector. With respect to the communication cost in Query, the size of the trapdoor is the same as that of the subindex listed in the Table 3, which 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 [16], [18], our proposed schemes introduce nearly constant overhead while increasing the number of query keywords. Therefore, our schemes cannot be compromised by timing-based side channel attacks that try to differentiate certain queries based on their query time.

## 6 Related Work

### 6.1 Single Keyword Searchable Encryption

Traditional single keyword searchable encryption schemes [7], [8], [9], [10], [11], [12], [13], [14], [15], [25], [26] 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) [4]. It is first studied by Song et al. [7] in the symmetric key setting, and improvements and advanced security definitions are given in Goh [8], Chang et al. [9], and Curtmola et al. [10]. Our early works [25], [26] solve secure ranked keyword search which utilizes keyword frequency to rank results instead of returning undifferentiated results. However, they only supports single keyword search. In the public key setting, Boneh et al. [11] 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.

### 6.2 Boolean Keyword Searchable Encryption

To enrich search functionalities, conjunctive keyword search [16], [17], [18], [19], [20] over encrypted data have been proposed. These schemes incur large overhead caused by their fundamental primitives, such as computation cost by bilinear map, for example, [18], or communication cost by secret sharing, for example, [17]. As a more general search approach, predicate encryption schemes [21], [22], [23] 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. Our early work [1] has been aware of this problem, and provides solutions to the multi-keyword ranked search over encrypted data problem. In this paper, we extend and improve more technical details as compared to [1]. We propose two new schemes to support more search semantics which improve the search experience of the MRSE scheme, and also study the dynamic operation on the data set and index which addresses some important yet practical considerations for the MRSE design. On a different front, the research on top-$k$ retrieval [31] in database community is also loosely connected to our problem. Besides, Cao et. al. proposed a privacy-preserving graph containment query scheme [36] which solves the search problem with graph semantics.

## 7 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 improved MRSE schemes to achieve various stringent privacy requirements in two different threat models. We also investigate some further enhancements of our ranked search mechanism, including supporting more search semantics, i.e., TF × IDF, and dynamic data operations. Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given, and experiments on the real-world data set show our proposed schemes introduce low overhead on both computation and communication.

In our future work, we will explore checking the integrity of the rank order in the search result assuming the cloud server is untrusted.

## Acknowledgments

This work was supported in part by the US National Science Foundation under grants CNS-1116939, CNS-1217889, CNS-1218085, and CNS-1262277, and an ECS grant CityU138513 from the Research Grants Council of Hong Kong.

## References

[1] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," *Proc. IEEE INFOCOM*, pp. 829-837, Apr, 2011.

[2] L.M. Vaquero, L. Rodero-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.

[3] N. Cao, S. Yu, Z. Yang, W. Lou, and Y. Hou, "LT Codes-Based Secure and Reliable Cloud Storage Service," *Proc. IEEE INFOCOM*, pp. 693-701, 2012.

[4] S. Kamara and K. Lauter, "Cryptographic Cloud Storage," *Proc. 14th Int'l Conf. Financial Cryptograpy and Data Security*, Jan. 2010.

[5] A. Singhal, "Modern Information Retrieval: A Brief Overview," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 35-43, Mar. 2001.

[6] I.H. Witten, A. Moffat, and T.C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images.* Morgan Kaufmann Publishing, May 1999.

[7] D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," *Proc. IEEE Symp. Security and Privacy*, 2000.

[8] E.-J. Goh, "Secure Indexes," *Cryptology ePrint Archive*, http://eprint.iacr.org/2003/216. 2003.

[9] Y.-C. Chang and M. Mitzenmacher, "Privacy Preserving Keyword Searches on Remote Encrypted Data," *Proc. Third Int'l Conf. Applied Cryptography and Network Security*, 2005.

[10] R. Curtmola, J.A. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06)*, 2006.

[11] D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," *Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2004.

[12] M. Bellare, A. Boldyreva, and A. ONeill, "Deterministic and Efficiently Searchable Encryption," *Proc. 27th Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '07)*, 2007.

[13] 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. Cryptology*, vol. 21, no. 3, pp. 350-391, 2008.

[14] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy Keyword Search Over Encrypted Data in Cloud Computing," *Proc. IEEE INFOCOM*, Mar. 2010.

[15] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W.E.S. III, "Public Key Encryption That Allows PIR Queries," *Proc. 27th Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '07)*, 2007.

[16] P. Golle, J. Staddon, and B. Waters, "Secure Conjunctive Keyword Search over Encrypted Data," *Proc. Applied Cryptography and Network Security*, pp. 31-45, 2004.

[17] L. Ballard, S. Kamara, and F. Monrose, "Achieving Efficient Conjunctive Keyword Searches over Encrypted Data," *Proc. Seventh Int'l Conf. Information and Comm. Security (ICICS '05)*, 2005.

[18] D. Boneh and B. Waters, "Conjunctive, Subset, and Range Queries on Encrypted Data," *Proc. Fourth Conf. Theory Cryptography (TCC)*, pp. 535-554, 2007.

[19] R. Brinkman, "Searching in Encrypted Data," PhD thesis, Univ. of Twente, 2007.

[20] Y. Hwang and P. Lee, "Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-User System," *Pairing*, vol. 4575, pp. 2-22, 2007.

[21] J. Katz, A. Sahai, and B. Waters, "Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products," *Proc. 27th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2008.

[22] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption," *Proc. 29th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '10)*, 2010.

[23] E. Shen, E. Shi, and B. Waters, "Predicate Privacy in Encryption Systems," *Proc. Sixth Theory of Cryptography Conf. Theory of Cryptography (TCC)*, 2009.

[24] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized Private Keyword Search over Encrypted Data in Cloud Computing," *Proc. 31st Int'l Conf. Distributed Computing Systems (ICDCS '10)*, pp. 383-392, June 2011.

[25] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," *Proc. IEEE 30th Int'l Conf. Distributed Computing Systems (ICDCS '10)*, 2010.

[26] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data," *IEEE Trans. Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1467-1479, Aug. 2012.

[27] W.K. Wong, D.W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN Computation on Encrypted Databases," *Proc. 35th ACM SIGMOD Int'l Conf. Management of Data (SIGMOD),* pp. 139-152, 2009.

[28] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," *Proc. IEEE INFOCOM,* 2010.

[29] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," *Proc. IEEE INFOCOM,* 2010.

[30] S. Zerr, E. Demidova, D. Olmedilla, W. Nejdl, M. Winslett, and S. Mitra, "Zerber: r-Confidential Indexing for Distributed Documents," *Proc. 11th Int'l Conf. Extending Database Technology (EDBT '08),* pp. 287-298, 2008.

[31] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k Retrieval from a Confidential Index," *Proc. 12th Int'l Conf. Extending Database Technology (EDBT '09),* pp. 439-449, 2009.

[32] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Cryptography from Anonymity," *Proc. IEEE 47th Ann. Symp. Foundations of CS,* pp. 239-248, 2006.

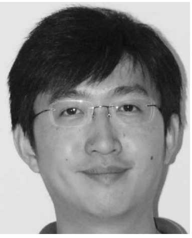
[33] J. Zobel and A. Moffat, "Exploring the Similarity Space," *ACM SIGIR Forum,* vol. 32, pp. 18-34, 1998.

[34] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward Secure and Dependable Storage Services in Cloud Computing," *IEEE Trans. Services Computing,* vol. 5, no. 2, pp. 220-232, Apr.-June 2012.

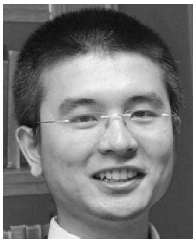
[35] W.W. Cohen, "Enron Email Data Set," http://www.cs.cmu.edu/~enron/, 2013.

[36] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Privacypreserving Query over Encrypted Graph-Structured Data in Cloud Computing," *Proc. Distributed Computing Systems (ICDCS),* pp. 393-402, June, 2011.

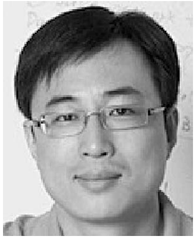
**Ning Cao** received the BE and ME degrees in computer science from Xi'an Jiaotong University, China, and the PhD degree in electrical and computer engineering from Worcester Polytechnic Institute. He is currently working at the Walmart Labs. He previously worked at the Research and System Infrastructure, Google Inc. His research interests include the areas of security, privacy, and reliability in Cloud Computing, with current focus on search and storage. He is a member of IEEE and a member of ACM.



**Cong Wang** received the BE and ME degrees from Wuhan University, and the PhD degree from Illinois Institute of Technology, all in electrical and computer engineering. He is an assistant professor in the Computer Science Department at City University of Hong Kong. He worked at the Palo Alto Research Center in the summer of 2011. His research interests include the areas of cloud computing security, with current focus on secure data outsourcing and secure computation outsourcing in public cloud. He is a member of the ACM and a member of the IEEE.



**Ming Li** (S'08—M'11) received the PhD degree in electrical and computer engineering from Worcester Polytechnic Institute, the ME and BE degrees in electronic and information engineering from Beihang University, China. He joined the Computer Science Department, Utah State University, as an assistant professor in 2011. His research interests include the general areas of cyber security and privacy, with current emphases on data security and privacy in cloud computing, security in wireless networks and cyber-physical systems. He is a member of IEEE and ACM.



**Kui Ren** received the PhD degree from Worcester Polytechnic Institute. He is currently an associate professor of computer science and engineering department at SUNY Buffalo. In the past, he has been an associate/assistant professor in the Electrical and Computer Engineering Department at Illinois Institute of Technology. His research interests include Cloud Security, Wireless Security, and Smartphone-enabled Crowdsourcing Systems. His research has been supported by the US National Science Foundation (NSF), the US Department of Energy (DoE), AFRL, and Amazon. He is a recipient of the NSF CAREER Award in 2011. He received the Best Paper Award from IEEE ICNP 2011. Kui serves as an associate editor for *IEEE Transactions on Information Forensics and Security*, *IEEE Wireless Communications*, *IEEE Transactions on Smart Grid*, *IEEE Internet of Things Journal*, *IEEE Communications Surveys and Tutorials*, *Elsevier Pervasive and Mobile Computing*, and the *Journal of Communications and Networks*. He is a senior member of the IEEE, a member of the ACM, and a past board member of Internet Privacy Task Force, State of Illinois. He is a senior member of the IEEE.

**Wenjing Lou** received the PhD degree in electrical and computer engineering, the University of Florida, 2003. She is currently an associate professor at Virginia Polytechnic Institute and State University. Prior to joining Virginia Tech in 2011, she was on the faculty of Worcester Polytechnic Institute from 2003 to 2011. Her current research interests include cyber security, with emphases on wireless network security and data security and privacy in cloud computing. She was a recipient of the US National Science Foundation CAREER award in 2008. She is a senior member of the IEEE.