Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data

Ning Cao[†], Cong Wang[‡], Ming Li[†], Kui Ren[‡], and Wenjing Lou[†]
[†]Department of ECE, Worcester Polytechnic Institute, Email: {ncao, mingli, wjlou}@ece.wpi.edu
[‡]Department of ECE, Illinois Institute of Technology, Email: {cong, kren}@ece.iit.edu

***Abstract*—With the advent of cloud computing, data owners are motivated to outsource their complex data management systems from local sites to 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 cloud, it is crucial for the search service to allow multi-keyword query and provide result similarity ranking to meet the effective data retrieval need. Related works on searchable encryption focus on single keyword search or Boolean keyword search, and rarely differentiate 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), and establish a set of strict privacy requirements for such a secure cloud data utilization system to become a reality. Among various multi-keyword semantics, we choose the efficient principle of "coordinate matching", i.e., as many matches as possible, to capture the similarity between search query and data documents, and further use "inner product similarity" to quantitatively formalize such principle for similarity measurement. We first propose a basic MRSE scheme using secure inner product computation, and then significantly improve it to meet different privacy requirements in two levels of threat models. Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given, and experiments on the real-world dataset further show proposed schemes indeed introduce low overhead on computation and communication.**

I. 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, especially when the data produced by them that need to be stored and utilized is rapidly increasing. To protect data privacy and combat unsolicited accesses in cloud and beyond, sensitive data, e.g., emails, personal health records, photo albums, tax documents, financial transactions, etc., may have to be encrypted by data owners before outsourcing to 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 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, large amount of documents demand cloud server to perform result relevance ranking, instead of returning undifferentiated result. 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 search result accuracy as well as enhance user searching experience, it is also crucial for such ranking system to support multiple keywords search, as single keyword search often yields far too coarse result. 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 principle 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 data privacy, index privacy, 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 over it through single keyword and retrieve documents of interest. However, direct application of these approaches to deploy secure large scale cloud data utilization system would not be necessarily suitable, as they are developed as crypto primitives and cannot accommodate such high service-level requirements like system usability, user searching experience, and easy information discovery in mind.

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 solves the secure ranked search over encrypted data with support of only single keyword query. But how to design an efficient encrypted data search mechanism that supports multi-keyword semantics without privacy breaches still remains an 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 cloud computing paradigm. Among various multi-keyword semantics, we choose the efficient principle of "coordinate matching", i.e., as many matches as possible, to capture the similarity between search query and data documents. Specifically, we use "inner product similarity" [4], i.e., the number of query keywords appearing in a document, to quantitatively evaluate the similarity of that document to the search query in "coordinate matching" principle. During 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 inner product of query vector with data vector. However, directly outsourcing data vector or query vector will violate index privacy or search privacy. To meet the challenge of supporting such multi-keyword semantic without privacy breaches, we propose a basic MRSE scheme using secure inner product computation, which is adapted from a secure k-nearest neighbor (*kNN*) technique [4], and then improve it step by step to achieve various privacy requirements in two levels of threat models. 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 to become a reality.
2) We propose two MRSE schemes following the principle of "coordinate matching" while meeting different privacy requirements in two levels of threat models.
3) Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given, and experiments on the real-world dataset further show 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 and threat model, our design goals, and preliminary. Section III describes MRSE framework and privacy requirements, followed by section IV, which gives our schemes achieving efficiency and privacy requirements. 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.

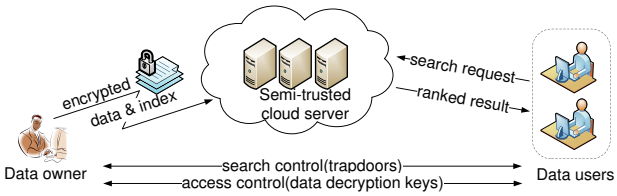


Fig. 1: Architecture of the search over encrypted cloud data

II. Problem Formulation

A. System Model

Considering a cloud data hosting service involving three different entities, as illustrated in Fig. 1: data owner, data user, and cloud server. Data owner has a collection of data documents $\mathcal{F}$ to be outsourced to cloud server in the encrypted form $\mathcal{C}$. To enable the searching capability over $\mathcal{C}$ for effective data utilization, 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 cloud server. To search the document collection for t given keywords, an authorized user acquires a corresponding trapdoor T through search control mechanisms, e.g., broadcast encryption [8]. Upon receiving T from data users, cloud server is responsible to search the index $\mathcal{I}$ and return the corresponding set of encrypted documents. To improve document retrieval accuracy, search result should be ranked by cloud server according to some ranking criteria (e.g., coordinate matching, as will be introduced shortly). Moreover, to reduce communication cost, data user may send an optional number k along with the trapdoor T so that cloud server only sends back top-k documents that are most relevant to the search query. Finally, the access control mechanism is employed to manage decryption capabilities given to users.

B. Threat Model

Cloud server is considered as "honest-but-curious" in our model, which is consistent with the most related works on searchable encryption. Specifically, 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 cloud server knows, we consider two levels of threat models as follows.

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

Known Background Model In this stronger model, cloud server is supposed to possess some backgrounds on the dataset, such as the subject and its related statistical information, in addition to what can be accessed in known ciphertext model. As an instance of possible attacks in this case, cloud server could utilize document frequency or keyword frequency [23] to identify keywords in the query.

C. 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 cloud server from learning additional information from dataset and index, and to meet privacy requirements specified in section III-B.
- **Efficiency:** Above goals on functionality and privacy should be achieved with low communication and computation overhead.

D. Notations

- $\mathcal{F}$ – the plaintext document collection, denoted as a set of m data documents $\mathcal{F} = (F_1, F_2, \ldots, F_m)$.
- $\mathcal{C}$ – the encrypted document collection stored in cloud server, denoted as $\mathcal{C} = (C_1, C_2, \ldots, C_m)$.
- $\mathcal{W}$ – the distinct keywords extracted from document collection $\mathcal{F}$, denoted as $\mathcal{W} = (W_1, W_2, \ldots, W_n)$.
- $\mathcal{I}$ – the searchable index associated with $\mathcal{C}$, denoted as $(I_1, I_2, \ldots, I_m)$ 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 $\widetilde{\mathcal{W}} = (W_{j_1}, W_{j_2}, \ldots, W_{j_t})$.
- $T_{\widetilde{\mathcal{W}}}$ – the trapdoor for the search request $\widetilde{\mathcal{W}}$.
- $\mathcal{F}_{\widetilde{\mathcal{W}}}$ – the ranked id list of all documents according to their similarity with $\widetilde{\mathcal{W}}$.

E. Preliminary on Coordinate Matching

As a hybrid of conjunctive search and disjunctive search, "coordinate matching" [4] is an intermediate approach which uses the number of query keywords appearing in the document to quantify the similarity 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 specify a list of keywords indicating their interest and retrieve the most relevant documents with 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 data owner could easily employ traditional symmetric key cryptography to encrypt and then outsource data. With focus on index and query, a MRSE consists of four algorithms as follows.

- $\mathsf{Setup}(1^\ell)$ *Taking a security parameter ℓ as input, data owner outputs a symmetric key as SK.*
- $\mathsf{BuildIndex}(\mathcal{F}, SK)$ *Based on the dataset $\mathcal{F}$, data owner builds a searchable index $\mathcal{I}$ which is encrypted by the symmetric key SK and then outsourced to cloud server. After the index construction, the document collection can be independently encrypted and outsourced.*
- $\mathsf{Trapdoor}(\widetilde{\mathcal{W}})$ *With t keywords of interest in $\widetilde{\mathcal{W}}$ as input, this algorithm generates a corresponding trapdoor $T_{\widetilde{\mathcal{W}}}$.*
- $\mathsf{Query}(T_{\widetilde{\mathcal{W}}}, k, \mathcal{I})$ *When 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 $\widetilde{\mathcal{W}}$.*

Both search control and access control are not 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.

B. 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*, data owner can resort to traditional symmetric key cryptography to encrypt the data before outsourcing, and successfully prevent cloud server from prying into outsourced data. With respect to the *index privacy*, if 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 [23]. Therefore, searchable index should be constructed to prevent 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 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, 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 [24]. When cloud server knows some background information of the dataset, this keyword specific information may be utilized to reverse-engineer the keyword.

Trapdoor Privacy Since only authorized users are allowed to acquire trapdoors for their search query, the server is not

expected to have the ability to generate valid trapdoors from previous received ones. Specifically, given one trapdoor for a set of multiple keywords, the server is not allowed to generate a valid trapdoor for its subset, including single keyword. For example, it is forbidden to generate or deduce a new trapdoor as T_{W_i} for keyword W_i from the received trapdoor as $T_{(W_i,W_k)}$ for two keywords (W_i, W_j). Moreover, the server is not allowed to generate a valid trapdoor, e.g., $T_{(W_i,W_j)}$, from two or more trapdoors, like $T_{(W_i,W_k)}$ and $T_{(W_j,W_k)}$.

Search Pattern In accordance with the definition in related work on single keyword searchable encryption [8], search pattern of data user in MRSE means any information that can be derived by server if it acquires the knowledge that two arbitrary searches are performed for the same keywords or not. If the trapdoor is generated in a deterministic manner, server could easily know the search pattern of any data user by comparing trapdoors received from that user. So the fundamental protection for search pattern is to introduce nondeterminacy into trapdoor generation procedure.

Access Pattern Within the ranked search, 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 $\widetilde{\mathcal{W}}$ is denoted as $\mathcal{F}_{\widetilde{\mathcal{W}}}$, consisting of the id list of all documents ranked by their similarity 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, e.g., [17] has been proposed to utilize private information retrieval(PIR) technique [25], to hide access pattern, our proposed schemes are not designed to protect access pattern for the efficiency concerns. This is because any PIR based technique must "touch" the whole dataset outsourced on the server which is inefficient in the large scale cloud system.

IV. Privacy-Preserving and Efficient MRSE

To efficiently achieve multi-keyword ranked search, we propose to employ "inner product similarity" [4] to quantitatively formalize the efficient ranking principle "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, 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 D_i, query vector Q and their inner product $D_i \cdot Q$ should not be exposed to cloud server. In this section, we first propose a basic MRSE scheme using secure inner product computation, which is adapted from a secure k-nearest neighbor (kNN) technique, and then show how to significantly improve it to be privacy-preserving against different levels of threat models in the MRSE framework in a step-by-step manner.

A. MRSE_I: Basic Scheme

1) Secure kNN Computation: In the secure k-nearest neighbor (kNN) scheme [26], Euclidean distance between a database 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. Without prior knowledge of secret key, neither data vector nor query vector, after such a series of processes, can be recovered by analyzing their corresponding ciphertext.

As MRSE is using inner product similarity instead of Euclidean distance, we need to do some modifications on the data structure to fit the MRSE framework. By eliminating dimension extension, the final result changes to be the inner product as $rp_i \cdot q$, and it seems that an efficient inner product computation scheme can be directly achieved. However, this approach hides the product only by a scale factor r which will leak the relationship among different queries. For example, if two queries are taking for the same keywords, denoted as rq and $r'q$, similarity scores in two queries will satisfy the scale relationship, i.e., $(p_i \cdot rq)/(p_i \cdot r'q) = (p_j \cdot rq)/(p_j \cdot r'q) = r/r'$. As a consequence, the *search pattern* of data user is leaked via examining whether similarity scores for all documents in two queries hold such relationship.

2) MRSE_I Scheme: To provide a guarantee against violation on search pattern clearly presented above, we have to eliminate the scale relationship among similarity scores in different queries. To do so, instead of simply removing the extended dimension as we plan to do at the first glance, we preserve this dimension extending operation but assign a random number to the extended dimension in each query vector. The whole scheme to achieve ranked search with multiple keywords over encrypted data is as follows.

- Setup After extracting the distinct keywords set $\mathcal{W}$ from the document collection $\mathcal{F}$, data owner randomly generates a $(n+1)$-bit vector as S and two $(n+1) \times (n+1)$ 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$(\mathcal{F}, SK)$ Data owner generates a binary data vector D_i for every document F_i, where each binary bit

$D_i[j]$ represents the existence of the corresponding keyword W_j in that document. Subsequently, every subindex I_i is generated by applying dimension extending, splitting and encrypting procedures on D_i. These procedures are similar with those above except that the $(n+1)$-th entry in $\vec{D}_i$ is set to 1 during dimension extending. Finally, subindex $I_i = \{M_1^T\vec{D}_i', M_2^T\vec{D}_i''\}$ is built for every encrypted document C_i on cloud server.

- Trapdoor($\widetilde{\mathcal{W}}$) With t keywords of interest in $\widetilde{\mathcal{W}}$ as input, one binary vector Q is first generated where each bit $Q[j]$ indicates whether $W_j \in \widetilde{\mathcal{W}}$ is true or false. Q is then scaled by a random number $r \neq 0$ as rQ, and extended to a $(n+1)$-dimension vector as $\vec{Q} = (rQ, t)$ where t is another random number. 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 $T_{\widetilde{\mathcal{W}}}$, cloud server computes the similarity scores of each document F_i as in equation 1. WLOG, we assume $r > 0$. After sorting all scores, cloud server returns the top-k ranked id list $\mathcal{F}_{\widetilde{\mathcal{W}}}$.

With the randomness t brought into the query vector, the final similarity scores do not keep the proportional relationship to the original inner product and therefore prevent cloud server from guessing search pattern through search results.

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

3) Analysis: We analyze this basic MRSE scheme from three aspects of design goals described in section II.

Functionality and Efficiency Assume the number of query keywords appearing in a document F_i is $x_i = D_i \cdot Q$. From equation 1, the final similarity score as $y_i = I_i \cdot T_{\widetilde{\mathcal{W}}} = rx_i + t$ is a linear function of x_i. Besides, the coefficient r is set as a positive random number, so the order of similarity is exactly preserved for all the outsourced documents. When cloud server creates the $\mathcal{F}_{\widetilde{\mathcal{W}}}$ by the final similarity score y_i, the top-k most similar documents to the query are included with the precise rank order. Similar with the secure kNN scheme, our inner product based MRSE scheme is an outstanding approach from the performance perspective. In the step like BuildIndex or Trapdoor, the generation procedure of each subindex or trapdoor involves two multiplications of a $(n+1) \times (n+1)$ matrix and a $(n+1)$-dimension vector. In the Query, the final similarity score is computed through two multiplications of two $(n+1)$-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 known ciphertext model [26]. In addition to the random number t in the query result, our basic scheme can generate two totally different trapdoors for the same query $\widetilde{\mathcal{W}}$. Therefore, the *search pattern* is well protected in our basic scheme, while it is an unsolved privacy leakage problem in related symmetric key based searchable encryption schemes because of the deterministic property of trapdoor generation.

TABLE I: Min/Max Score Analysis I

$r > 0$	$min\{y_j\} = r \cdot min\{x_i\} + t$
$r > 0$	$submin\{y_j\} = r \cdot submin\{x_i\} + t$
$r < 0$	$max\{y_j\} = r \cdot max\{x_i\} + t$
$r < 0$	$submax\{y_j\} = r \cdot submax\{x_i\} + t$

B. MRSE_II: Privacy-Preserving Scheme in Known Ciphertext Model

MRSE_I scheme performs outstanding from the efficiency perspective and also provides privacy guarantee on search pattern, but it will incur trapdoor privacy leakage once cloud server is requested by users to execute two or more times of searches. By analyzing similarity scores obtained during search, cloud server has a chance to deduce a valid trapdoor which violates trapdoor privacy goal. We first demonstrate how such analysis attack works, and then show this problem can be fixed through inserting dummy keyword.

1) Min/Max Score Analysis Attack: With any two valid trapdoors T_1 and T_2 submit by users, cloud server may explore the relationship among similarity scores to deduce a new trapdoor T_3. If T_1 and T_2 happen to be trapdoors for two related sets of query keywords, like $\{K_1, K_2\}$ and $\{K_1, K_2, K_3, K_4\}$, T_3 then becomes a valid trapdoor for keywords $\{K_3, K_4\}$. To illustrate, the relationship between final similarity score y_j with the original one x_i is listed in Tab. I. WLOG, we only discuss the case where $r > 0$. In the query Q_1 with the trapdoor $T_1 = \{T_1[1], T_1[2]\}$, there may exist a document containing neither K_1 nor K_2 with very high probability. As a result, the minimal original similarity score as $min\{x_i\}$ is equal to 0 and the minimal final similarity score as $min\{y_j\}$ becomes t_1. Considering the large number of outsourced documents, it is very likely that there also exists a document containing only one of the two keywords, and then the subminimal original similarity score as $submin\{x_i\}$ is equal to 1 and the subminimal final similarity score as $submin\{y_j\}$ becomes $r_1 + t_1$. With t_1 and $r_1 + t_1$, cloud server solves r_1 with apparent ease. Similarly, the other two parameters r_2 and t_2 can be figured out with $T_2 = \{T_2[1], T_2[2]\}$ for query Q_2. Assume that the original query vector for $\{K_3, K_4\}$ is denoted as Q_3. Note that Q_3 is equal to $Q_2 - Q_1$ according to the definition of binary query vector and the relationship between three keyword sets. Then the 2-tuple as $\{T_2[1]/r_2 - T_1[1]/r_1, T_2[2]/r_2 - T_1[2]/r_1\}$ can be utilized by cloud server as a valid trapdoor T_3 for query Q_3 where r_3 is set to 1 and t_3 is set to $(r_2/r_2 - t_1/r_1)$. The effectiveness of T_3 is validated in equation 2.

$$\begin{aligned} & I_i \cdot T_3 \\ =\ & I_i \cdot \{T_2[1]/r_2 - T_1[1]/r_1, T_2[2]/r_2 - T_1[2]/r_1\} \\ =\ & \frac{I_i[1] \cdot T_2[1] + I_i[2] \cdot T_2[2]}{r_2} - \frac{I_i[1] \cdot T_1[1] + I_i[2] \cdot T_1[2]}{r_1} \\ =\ & (\vec{D}_i \cdot \vec{Q}_2)/r_2 - (\vec{D}_i \cdot \vec{Q}_1)/r_1 \\ =\ & (D_i \cdot Q_2 + t_2/r_2) - (D_i \cdot Q_1 + t_1/r_1) \\ =\ & r_3(D_i \cdot Q_3) + t_3 \end{aligned} \quad (2)$$

Actually, such kind of trapdoor deduction enables cloud server to perform query that is not requested by any user, and thus

TABLE II: Min/Max Score Analysis II

$r > 0$	$min\{y_j\} = r \cdot min\{x_i + \varepsilon_i\} + t$
$r > 0$	$submin\{y_j\} = r \cdot submin\{x_i + \varepsilon_i\} + t$
$r < 0$	$max\{y_j\} = r \cdot max\{x_i + \varepsilon_i\} + t$
$r < 0$	$submax\{y_j\} = r \cdot submax\{x_i + \varepsilon_i\} + t$

violates search privacy, specifically the trapdoor privacy.

2) MRSE_II Scheme: The previous analysis attack shows that the trapdoor privacy problem stems from the deterministic relationship between the minimal/subminimal (or maximal/submaximal) final similarity score and two parameters as r and t. Therefore, breaking the deterministic relationship is an alternative to protect trapdoor privacy. In this section, we propose an improved MRSE scheme preserving trapdoor privacy in the known ciphertext model. Namely, we will show how to break the deterministic relationship while maintaining as much efficiency and accuracy as possible. 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. All the vectors are extended to $(n+2)$-dimension instead of $(n+1)$, and each entry in D_i is not a binary value anymore for storing the random variable ε_i of the dummy keyword in the $(n+2)$-th dimension. Improvement details in MRSE_II scheme is shown as follows.

- Setup Data owner randomly generates a $(n+2)$-bit vector as S and two $(n+2) \times (n+2)$ invertible matrices as $\{M_1, M_2\}$.
- BuildIndex$(\mathcal{F}, SK)$ The $(n+2)$-th entry in $\vec{D_i}$ is set to a random number ε_i during the dimension extending.
- Trapdoor$(\widetilde{\mathcal{W}})$ The $(n+2)$-th entry in $\vec{Q}$ is always set to 1 for any query Q.
- Query$(T_{\widetilde{\mathcal{W}}}, k, \mathcal{I})$ The final similarity score computed by cloud server is equal to $r(x_i + \varepsilon_i) + t_i$.

3) Analysis: We follow the same steps as in MRSE_I.

Functionality and Efficiency Because the randomness 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 MRSE_I scheme. For the consideration of search accuracy, let ε follow a normal distribution $N(0, \sigma^2)$. 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 dataset will be given in section VI. Besides, MRSE_II scheme takes similar computation and communication costs as in MRSE_I scheme, except that all the vector multiplication operations are run in the $(n+2)$-dimension instead of $(n+1)$-dimension.

Privacy We first take a look at *trapdoor privacy*, especially the Min/Max score analysis. Similar with the case discussed above, we assume that cloud server has two trapdoors T_1 and T_2 for query keywords $\{K_1, K_2\}$ and $\{K_1, K_2, K_3, K_4\}$ respectively. To perform Min/Max score analysis, the server scans the entire index and computes all the final similarity scores as listed in Tab. II. With the interference of ε, the adversary cannot explore useful relationship between the minimal/subminimal final similarity score and two parameters as r and t, and thus the Min/Max score analysis attack does not work anymore. In addition, any single trapdoor itself does not reveal valuable information, like the positions of 1 in its original query vector Q, and therefore keyword privacy is well protected. To sum up, in the known ciphertext model, MRSE_II scheme meets all expected privacy requirements mentioned in section III-B.

TABLE III: 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	$x_2 = 2, y_2 = r(2+\varepsilon_2) + t$	$x'_2 = 1, y'_2 = r'(1+\varepsilon_1) + t'$
3	$x_3 = 1, y_3 = r(1+\varepsilon_3) + t$	$x'_3 = 0, y'_3 = r'(0+\varepsilon_3) + t'$

TABLE IV: 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	$x_2 = 1, y_2 = r(1+\varepsilon_2) + t$	$x'_2 = 1, y'_2 = r'(1+\varepsilon_1) + t'$
3	$x_3 = 0, y_3 = r(0+\varepsilon_3) + t$	$x'_3 = 0, y'_3 = r'(0+\varepsilon_3) + t'$

C. MRSE_III: Privacy-Preserving Scheme in Known Background Model

When cloud server has known some background information on the outsourced dataset, keyword privacy cannot be guaranteed anymore by MRSE_II scheme. This is possible in known background model because cloud server can use scale analysis as follows to deduce the keyword-specific information, e.g., document frequency, which can be further combined with background information to identify the keyword in a query at high probability. After presenting how cloud server uses scale analysis attack to break keyword privacy, we propose a more advanced MRSE scheme to be privacy-preserving in known background model.

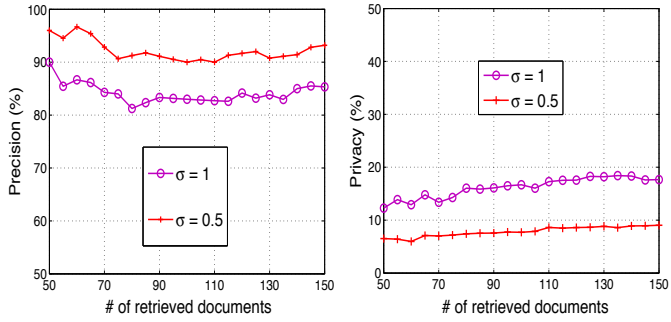
1) Scale Analysis Attack: Given two correlated trapdoors T_1 and T_2 for query keywords $\{K_1, K_2\}$ and $\{K_1, K_2, K_3\}$ respectively, there will be two special cases when searching on any three documents as listed in Tab. III and Tab. IV. In any of these two cases, there exists a system of equations among those similarity scores as follows,

$$\begin{cases} y_1 - y_2 = \quad r(1 + \varepsilon_1 - \varepsilon_2); \\ y'_1 - y'_2 = \quad r'(1 + \varepsilon_1 - \varepsilon_2); \\ y_1 - y_3 = \quad r(2 + \varepsilon_1 - \varepsilon_2); \\ y'_1 - y'_3 = \quad r'(2 + \varepsilon_1 - \varepsilon_2). \end{cases} \tag{3}$$

And cloud server could deduce the following scale relationship among all the six scores being consistent with equation 3,

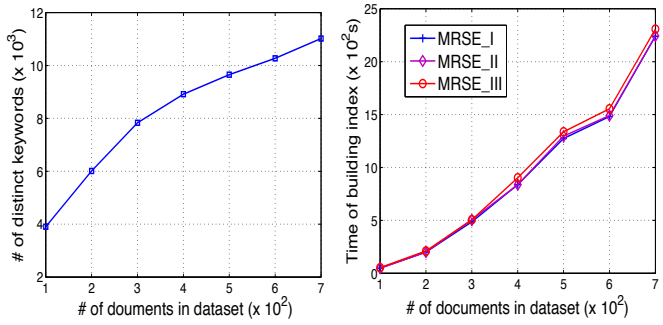
$$(y_1 - y_2)/(y'_1 - y'_2) = (y_1 - y_3)/(y'_1 - y'_3). \tag{4}$$

To this end, although the exact value of x_i is encrypted as y_i, cloud server, based on the equivalence of $(y_1 - y_2)/(y'_1 - y'_2)$ and $(y_1 - y_3)/(y'_1 - y'_3)$, could deduce that whether all the three documents contain K_3 or none of them contain K_3. By extending three documents to the whole dataset, cloud server could deduce two possible values of document frequency of the keyword K_3. In known background model, the server can further identify the keyword K_3 by referring to the keyword specific document frequency information about the dataset.



(a) (b)

Fig. 2: With different choice of standard deviation σ for the random variable ε, there exists tradeoff between (a) Precision, and (b) Rank Privacy.



(a) (b)

Fig. 3: Relationship between number of documents in dataset and (a) Number of distinct keywords in dataset, and (b) Time cost for building searchable index.

2) MRSE_III Scheme: The privacy leakage shown above is caused by the fixed value of random variable ε_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 P_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 MRSE_III scheme is presented as follows.

- $\mathsf{Setup}(1^n)$ Data owner randomly generates a $(n+U+1)$-bit vector as S and two $(n+U+1)\times(n+U+1)$ invertible matrices $\{M_1, M_2\}$.
- $\mathsf{BuildIndex}(\mathcal{F}, SK)$ 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.
- $\mathsf{Trapdoor}(\widetilde{\mathcal{W}})$ By randomly selecting V out of U dummy keywords, the corresponding entries in Q are set to 1.
- $\mathsf{Query}(T_{\widetilde{\mathcal{W}}}, k, \mathcal{I})$ 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.

3) Analysis: To achieve the 80-bit security level, there should be at least 2^{80} different values of $\sum \varepsilon_i^{v)}$ for each data vector. The number of different $\sum \varepsilon_i^{(v)}$ is equal to $\binom{U}{V}$, which is maximized when $\frac{U}{V} = 2$. Besides, considering $\binom{U}{V} \geq (\frac{U}{V})^V$, it is greater than 2^{80} when $U = 160$ and $V = 80$. So every data vector should include 160 dummy elements, and every query vector will randomly select 80 dummy elements. To this end, MRSE_III scheme is secure against scale analysis attack, and provides various expected privacy guarantees within known ciphertext model or known background model.

Moreover, to make $\sum \varepsilon_i^{(v)}$ follow the Normal distribution $N(0, \sigma^2)$ as above, every $\varepsilon^{(j)}$ is assumed to follow the same uniform distribution $M(-c, c)$, where the mean as μ_j is 0 and the variance as σ_j^2 is $c^2/3$. According to the central limit theorem, the sum of 80 independent random variables $\varepsilon^{(j)}$ follows the Normal distribution, where $\mu = 80\mu_j$=0 and $\sigma^2 = 80\sigma_j^2 = 80c^2/3$. Therefore, the value of c is set as $\sqrt{\frac{3}{80}}\sigma$. With such parameter setting, search accuracy is statistically the same as that in MRSE_II scheme.

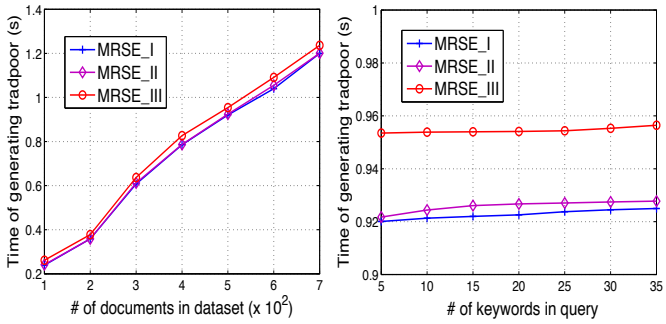
V. Performance Analysis

In this section, we demonstrate a thorough experimental evaluation of the proposed technique on a real-world dataset: the Enron Email Dataset [27]. 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 three proposed MRSE schemes, as well as the tradeoff between search precision and privacy.

A. Precision and Privacy

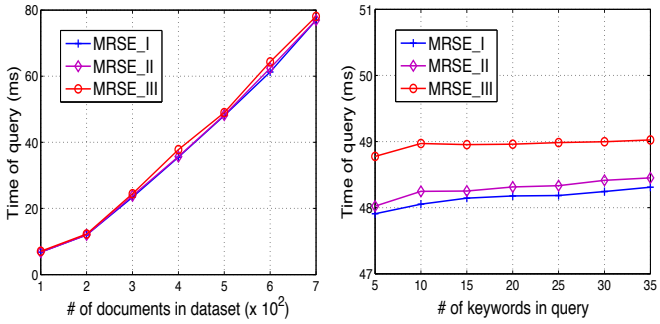
As presented in Section IV, dummy keywords are inserted into each data vector and some of them are selected in every query. Therefore, similarity scores of documents will be not exactly accurate. In other words, when cloud server returns top-k documents based on similarity scores of data vectors to query 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 cloud server. Fig. 2(a) shows that the precision in MRSE_III scheme is evidently affected by the standard deviation σ of the random variable ε. 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 privacy may have been partially leaked to cloud server as a consequence of small σ. As described in section III-B, 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 perturbation as $\widetilde{p}_i = |r_i - r'_i|$, where r_i is the rank number of document i in the retrieved top-k documents and r'_i is its rank number in the real top-i ranked documents. The overall rank



(a) (b)

Fig. 4: Time cost of generating trapdoor. (a) For the same number (10) of keywords in a query within different number of documents in dataset. (b) For different number of keywords in a query within same number (500) of documents in dataset.



(a) (b)

Fig. 5: Time cost of query. (a) For the same number (10) of keywords in a query within different number of documents in dataset. (b) For different number of keywords in a query within the same number (500) of documents in dataset.

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. 2(b) 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 σ leads to higher precision of search result but lower rank privacy guarantee, while large σ 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.

B. Efficiency

1) Index Construction: To build a searchable index $\mathcal{I}$ from dataset $\mathcal{F}$, the first step is to map the keyword set extracted from each 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 number of distinct keywords in the dataset. Fig. 3(a) shows that the number of documents in dataset determines the number of distinct keywords. Note that a list of standard IR techniques can be used to significantly reduce the number of distinct keywords, such as case folding, stemming, and stop words. We omit this refining process and refer readers to [4] for more details. Fig. 3(b) shows that the computation cost of building index is almost linear with the number of documents in dataset. The index construction computation cost in MRSE_I scheme is very similar with that in MRSE_II scheme since the dimensionality in MRSE_II is just one more than that in the former scheme. The number of dummy keywords in MRSE_III scheme becomes 160 so that the index construction time is slight larger than the other two schemes. Although the time of building index is not a negligible overhead for data owner, this is a one-time operation before data outsourcing. Besides, Tab. V lists the storage overhead of subindex in MRSE_III scheme within different number of documents in dataset. The subindex size is absolutely linear with the dimensionality of data vector, but its increasing speed slows down in coincidence with that of the number of distinct keywords. The subindex size in the other two MRSE schemes is close to that in MRSE_III scheme because of trivial differences in dimensionality of data vector.

TABLE V: Storage overhead of subindex

# of documents in dataset	1000	2000	3000	4000	5000
Size of subindex (KB)	101.4	131.8	166.6	203.8	228.9

2) Trapdoor Generation: Fig. 4(a) shows that the time to generate a trapdoor is greatly affected by the number of documents in dataset. 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 three proposed schemes and becomes larger with the increasing number of documents in dataset. Fig. 4(b) demonstrates the trapdoor generation cost in MRSE_III scheme is about 3 percentages larger than that in the other two schemes, which is majorally brought by the larger dimensionality. More importantly, it shows that the number of keywords in a query have 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 cloud server consists of computing and ranking similarity scores for all documents in the dataset. Fig. 5 shows the query time is dominated by the number of documents in dataset, and the number of keywords in the query has very slight impact on it like the trapdoor generation above. With respect to the communication cost in Query, the size of trapdoor is the same as that of subindex listed in the Tab. V, which keeps constant in the same dataset, 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 enjoy the constant overhead in the query which makes it more practical in the cloud paradigm.

VI. Related Work

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.

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 cloud. On a different front, the research on top-k retrieval [24] 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 principle of "coordinate matching", i.e., as many matches as possible, to effectively capture similarity between query keywords and outsourced documents, and use "inner product similarity" to quantitatively formalize such a principle for similarity measurement. For meeting the challenge of supporting multi-keyword semantic without privacy breaches, we first propose a basic MRSE scheme using secure inner product computation, and significantly improve it to achieve privacy requirements in two levels of 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 more stronger threat model.

References

[1] 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.

[2] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *RLCPS, January 2010, LNCS. Springer, Heidelberg*.

[3] A. 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: Compressing 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. ONeill, "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.

[21] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proc. of TCC*, 2009.

[22] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. of ICDCS'10*, 2010.

[23] S. Zerr, E. Demidova, D. Olmedilla, W. Nejdl, M. Winslett, and S. Mitra, "Zerber: r-confidential indexing for distributed documents," in *Proc. of EDBT*, 2008, pp. 287–298.

[24] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k retrieval from a confidential index," in *Proc. of EDBT*, 2009.

[25] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Cryptography from anonymity," in *Proc. of FOCS*, 2006, pp. 239–248.

[26] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. of SIGMOD*, 2009.

[27] W. W. Cohen, "Enron email dataset," http://www.cs.cmu.edu/~enron/.