Multi-keyword Similarity Search over Encrypted Cloud Data*

Mikhail Strizhov and Indrajit Ray

Colorado State University
Fort Collins, CO, 80523, USA
{strizhov,indrajit}@CS.ColoState.EDU

Abstract. Searchable encryption allows one to upload encrypted documents on a remote *honest-but-curious* server and query that data at the server itself without requiring the documents to be decrypted prior to searching. In this work, we propose a novel secure and efficient multi-keyword similarity searchable encryption (MKSim) that returns the matching data items in a ranked ordered manner. Unlike all previous schemes, our search complexity is sublinear to the total number of documents that contain the queried set of keywords. Our analysis demonstrates that proposed scheme is proved to be secure against adaptive chosen-keyword attacks. We show that our approach is highly efficient and ready to be deployed in the real-world cloud storage systems.

Keywords: searchable encryption, secure cloud outsourcing, multi-keyword ranked search, homomorphic encryption.

1 Introduction

Cloud computing enables new types of services where the computational and network resources are available online through the Internet. One of the most popular services of cloud computing is data outsourcing. For reasons of cost and convenience, public as well as private organizations can now outsource their large amounts of data to the cloud and enjoy the benefits of remote storage. At the same time, confidentiality of remotely stored data on untrusted cloud server is a big concern. In order to reduce these concerns, sensitive data, such as, personal health records, emails, income tax and financial reports, etc. are usually outsourced in encrypted form using well-known cryptographic techniques. Although encrypted data storage protects remote data from unauthorized access, it complicates some basic, yet essential data utilization services such as plaintext keyword search. A simple solution of downloading the data, decrypting and searching locally is clearly inefficient since storing data in the cloud is meaningless unless it can be easily searched and utilized. Thus, cloud services should enable efficient search on encrypted data to provide the benefits of a first-class cloud computing environment.

Researchers have investigated this problem quite extensively in the context of encrypted documents [1–6, 9, 10, 12, 13, 16, 20, 23, 25, 26]. Solutions generally involve

* This work was partially supported by the U.S. National Science Foundation under Grant No. 0905232

N. Cuppens-Boulahia et al. (Eds.): SEC 2014, IFIP AICT 428, pp. 52–65, 2014.

building an encrypted searchable index such that its content is hidden from the remote server yet allowing the corresponding documents to be searched. These solutions differ from each other mostly in terms of whether they allow single keyword search or multi-keyword search and the types of techniques they use to build the trapdoor function. A few of them, most notably [4,5,9,25], allow the notion of similarity search. The similarity search problem consists of a collection of data items that are characterized by some features, a query that specifies a value for a particular feature, and a similarity metric to measure the relevance between the query and the data items. However, these techniques either do not allow searching on multiple keywords and ranking the retrieved document in terms of similarity scores or are very computationally intensive. Moreover, none of these schemes are protected against *adaptive* adversaries [12]. Taking into account large volumes of data available today, there is need in efficient methods to perform secure similarity search over encrypted data outsourced into the cloud. In this work, we propose a novel secure and efficient multi-keyword similarity searchable encryption scheme that returns the matching data items in a ranked order.

Our contributions can be summarized as follows:

1. We present a secure searchable encryption scheme that allows multi-keyword query over an encrypted document corpus and retrieves the relevant documents ranked based on a similarity score.
2. We construct the searchable encryption scheme that is CKA2-secure in the random oracle model [12, 17]. Our scheme achieves semantic security against *adaptive* adversaries that choose their search queries as a function of previously obtained trapdoors and search outcomes.
3. We present a construction that achieves the *optimal* search time. Unlike all previous schemes that are glued to the linear search complexity, our search is sublinear to the total number of documents that contain the queried set of keywords. We show that this type of searchable encryption scheme can be extremely efficient.

The rest of the paper is organized as follows: Section 2 gives an outline of the most recent related work. In Section 3 we discuss the threat model for our problem space, give an overview of the system model, notations and preliminaries and introduce the building blocks used in our solution. In Section 4, we provide the framework of the proposed searchable encryption scheme and define the necessary security terms and requirements. The security analysis and comparison to other existing schemes is given in Section 5. Finally, Section 6 is devoted for the concluding remarks.

2 Related Work

Searchable encryption has been an active research area and many quality works have been published [1–6,9–13,16,20–23,25,26]. Traditional searchable encryption schemes usually build an encrypted searchable index such that its content is hidden to the server, however it still allows performing document searching with given search query. Song *et al.* [23] were the first to investigate the techniques for keyword search over encrypted and outsourced data. The authors begin with idea to store a set of plaintext documents on data storage server such as mail servers and file servers in encrypted form to reduce security and privacy risks. The work presents a cryptographic scheme that enables

indexed search on encrypted data without leaking any sensitive information to the untrusted remote server. Goh [16] developed a per-file Bloom filter-based secure index, which reduce the searching cost proportional to the number of files in collection. Recent work by Moataz *et al.* [21] proposed boolean symmetric searchable encryption scheme. Here, the scheme is based on the orthogonalization of the keywords according to the Gram-Schmidt process. Orencik's solution [22] proposed privacy-preserving multi-keyword search method that utilizes minhash functions. Boneh *et al.* [6] developed the first searchable encryption using the asymmetric settings, where anyone with the public key can write to the data stored remotely, but the users with private key execute search queries. The other asymmetric solution was provided by Di Crescenzo *et al.* in [11], where the authors propose a public-key encryption scheme with keyword search based on a variant of the quadratic residuosity problem.

All secure index based schemes presented so far, are limited in their usage since they support only exact matching in the context of keyword search. Wang *et al.* [25] studied the problem of secure ranked keyword search over encrypted cloud data. The authors explored the statistical measure approach that embeds the relevance score of each document during the establishment of searchable index before outsourcing the encrypted document collection. The authors propose a single keyword searchable encryption scheme using ranking criteria based on keyword frequency that retrieves the best matching documents. Cao *et al.* [9] presented a multi-keyword ranked search scheme, where they used the principle of "coordinate matching" that captures the similarity between a multi-keyword search query and data documents. However, their index structure is uses a binary representation of document terms and thus the ranked search does not differentiate documents with higher number of repeated terms than documents with lower number of repeated terms.

3 Background and Building Blocks

3.1 System and Threat Model

Consider a cloud data hosting service that involves three entities: data owner, cloud server and data user. The data owner may be an individual or an enterprise, who wishes to outsource a collection of documents $\mathsf{D} = (\mathsf{D}_1, \mathsf{D}_2, \ldots, \mathsf{D}_n)$ in encrypted form $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_n)$ to the cloud server and still preserve the search functionality on outsourced data. $\mathsf{C}_i = \mathsf{E}_S[\mathsf{D}_i]$ is the encrypted version of the document D_i computed using a semantically secure encryption scheme E with a secret key S. To enable multi-keyword ranked search capability, the data owner constructs searchable index I that is built on m distinct keywords $\mathsf{K} = (k_1, k_2, \ldots, k_m)$ extracted from the original dataset D. Both I and C are outsourced to the cloud server. To securely search the document collection for one or more keywords $\bar{\mathsf{K}} \in \mathsf{K}$, the authorized data user uses *search trapdoor* (distributed by the data owner) that generates the search request to the cloud server. Once the cloud server receives such request, it performs a search based on the stored index I and returns a ranked list of encrypted documents $\mathsf{L} \subseteq \mathsf{C}$ to the data user. The data user then uses the secret key *S*, securely obtained from the data owner, to decrypt received documents L to original view.

We assume a *honest-but-curious* model for the cloud server. The cloud server is *honest*, that is, it is always available to the data user and it correctly follows the designated protocol specification, and it provides all services that are expected. The *curious* cloud server may try to perform some additional analysis to breach the confidentiality of the stored data. In the rest of the paper, the cloud server and the adversary are the same entity. That way, the adversary has access to the same set of information as the cloud server. For this work, we are not concerned about the cloud server being able to link a query to a specific user; nor are we concerned about any denial-of-service attacks.

3.2 Notations and Preliminaries

Let $D = (D_1, D_2, \ldots, D_n)$ be a set of documents and $K = (k_1, k_2, \ldots, k_m)$ be the dictionary consisting of unique keywords in all documents in D, where $\forall\, i \in [1,m]\ k_i \in \{0,1\}^*$. $C = \{C_1, C_2, \ldots, C_n\}$ is an encrypted document collection stored in the cloud server. I_i is a searchable index associated with the corresponding encrypted document C_i. If A is an algorithm then $a \leftarrow A(\ldots)$ represents the result of applying the algorithm A to given arguments. Let R be an operational ring, we write vectors in bold, e.g. $\mathbf{v} \in R$. The notation $\mathbf{v}[i]$ refers to the i-th coefficient of $\mathbf{v}$. We denote the dot product of $\mathbf{u}$, $\mathbf{v} \in R$ as $\mathbf{u} \otimes \mathbf{v} = \sum_{i=1} \mathbf{u}[i] \cdot \mathbf{v}[i] \in R$. We use $\lfloor x \rceil$ to indicate rounding x to the nearest integer, and $\lfloor x \rfloor$, $\lceil x \rceil$ (for $x \geqslant 0$)to indicate rounding down or up.

Cryptographic Notations. A private-key encryption scheme is a set of three polynomial-time algorithms $SKE = (Gen, Enc, Dec)$ such that Gen is a probabilistic algorithm that takes a security parameter k and returns a secret key K_{secret}; Enc is a probabilistic algorithm that takes a key K_{secret} and a message m, and outputs a ciphtertext ξ; Dec is a deterministic algorithm that takes a secret key K_{secret} and a ciphertext ξ, and outputs m if K_{secret} is the valid secret key. We say that SKE is CPA-secure if the ciphtertexts it outputs do not reveal any partial information about the original plaintext to an adversary that can adaptively query an encryption oracle. We also make use of use pseudo-random function (PRF), which is a polynomial-time computable function that cannot be distinguished from random functions by any probabilistic polynomial-time adversary. We refer the reader [12, 17, 19] for formal definitions of *semantic security*, CPA-security and PRFs.

We now review definitions related to homomorphic encryption. Our definitions are based on Gentry's works [14] and [15], but we slightly relax the definition of decryption correctness, to allow a negligible probability of error.

Definition 1. ***Homomorphic encryption:*** *A homomorphic encryption scheme* Hom *consist of four algorithms:*

- $KeyGen$*: Given security parameter* λ*, returns a secret key* sk *and a public key* pk.
- Enc*: Given the plaintext* $m \in \{0,1\}$ *and public key* pk*, returns ciphertext* ϕ.
- Dec*: Given the ciphertext* ϕ *and secret key* sk*, returns the plaintext* m.
- $Eval$*: Given public key* pk*, a* t*-input circuit* C *(consisting of addition and multiplication gates modulo* 2*), and a tuple of ciphtertext* $(\phi_1, \phi_2, \ldots, \phi_t)$ *(corresponding to the* t *input bits of* C*), returns a ciphertext* ϕ.

Hom *is correct for a family* $\mathbb{C}$ *of circuits with* $\leqslant t = \mathsf{Poly}(\lambda)$ *input bits if for any* $C \in \mathbb{C}$ *and input bits* $(m_i)_{i \leqslant t}$*, the following holds with overwhelming probability over the randomness of* KeyGen *and* Enc*:*

$$\mathsf{Dec}(sk, \mathsf{Eval}(pk, C, (\phi_1, \phi_2, \ldots, \phi_t))) = C(m_1, m_2, \ldots, m_t), \tag{1}$$

where $(sk, pk) = \mathsf{KeyGen}(\lambda)$ *and* $\phi_i = \mathsf{Enc}(pk, m_i)$ *for* $i = 1, \ldots, t$.

Hom *is compact if for any circuit* C *with* $\leqslant t = \mathsf{Poly}(\lambda)$ *input bits, the bit-size of the ciphertext* $\mathsf{Eval}(pk, C, (\phi_1, \phi_2, \ldots, \phi_t))$ *is bounded by a fixed polynomial* $b(\lambda)$.

3.3 Brakerski's Homomorphic Cryptosystem

Brakerski *et al.* [7, 8] recently proposed encryption schemes that efficiently support low-degree homomorphic computations needed for our constructions. We use the ring-LWE-based variant of Brakerski's homomorphic cryptosystem [7] that operates over polynomial rings modulo a cyclotomic polynomial. One appealing property of Brakerski's homomorphic cryptosystem is to use the "batching mode" where a single ciphertext represent a vector of encrypted values and single homomorphic operation on two such ciphertexts applies the homomorphic operation component-wise to the entire vector. In such way, for the cost of a single homomorphic operation we get the compute on a entire vector of encrypted plaintexts. Let $\Delta_m(x)$ be the m^{th} cyclotomic polynomial and let $R = \mathbb{Z}[x]/\Delta_m(x)$ denote the operational ring. Moreover, let $R_p = \mathbb{Z}_p[x]/\Delta_m(x)$ be our plaintext space and let $R_q = \mathbb{Z}_q[x]/\Delta_m(x)$ be our ciphertext space for some $q > p$. Here, the secret keys are vectors of elements in R_q and the plaintext comprises of elements of R_p. We present the necessary basics of ring-LWE-based homomorphic encryption scheme: key-generation, encryption and decryption mechanisms:

$\underline{\mathsf{Hom.KeyGen}}$: We sample a polynomial $\bar{s} \in R_q$ from a Hamming weight distribution $\mathbb{HWT}(h)$. Here, h specifies the number of nonzero coefficients in $\bar{s}$ and each nonzero element in $\bar{s}$ is ± 1 with equal probability. The vector $\mathbf{s} = (1, \bar{s}) \in R_q^2$ is the secret key. The public key is generated by sampling uniformly a polynomial $\mathbf{A}$ from R_q and $\mathbf{e}$ from a noise distribution. The noise distribution is set to be the zero-mean discrete Gaussian distribution $\mathbb{DG}(\sigma^2)$ with variance σ^2. The public key is $\mathbf{P} = [\mathbf{b} || -\mathbf{A}] \in R_q^2$, where $\mathbf{b} = [\mathbf{A} \cdot \bar{s} + \mathbf{e}]_q$.

$\underline{\mathsf{Hom.Enc}(\mathbf{P}, \mathbf{m})}$: The algorithm that encrypts the message $\mathbf{m} \in R_p$ with public key $\mathbf{P}$. We sample a polynomial $\mathbf{r}$ with coefficients varying in $\{0, \pm 1\}$ and $\mathbf{e} = (\mathbf{e_0}, \mathbf{e_1})$ from the noise distribution $\mathbb{DG}^2(\sigma^2)$. The encryption of message $\mathbf{m}$ using public key $\mathbf{P}$ is a vector of ciphtertexts $\mathbf{c} = \left[\mathbf{P}^T\mathbf{r} + \left\lfloor \frac{q}{p} \right\rfloor \mathbf{m}\right]_q$, where $\lfloor \cdot \rfloor$ is the floor function and $[\cdot]_q$ denotes reduction modulo q.

$\underline{\mathsf{Hom.Dec}(\mathbf{S}, \mathbf{c})}$: The algorithm that decrypts a ciphertext $\mathbf{c}$ using the secret key $\mathbf{s}$ as follows: $m = \left[\left\lfloor p \cdot \frac{[\langle \mathbf{c}, \mathbf{s} \rangle]_q}{q} \right\rceil\right]_p$.

$\underline{\text{Addition}}$: $\mathbf{c_{sum}} = \mathbf{c_1} + \mathbf{c_2}$. E_{sum} is the result noise in $\mathbf{c_{sum}}$ and $E_{sum} \leqslant E_1 + E_2$, where E_1 and E_2 denote noise for $\mathbf{c_1}$ and $\mathbf{c_2}$.

Multiplication: $\mathbf{c_{prod}} = \left\lfloor \frac{p}{q} \cdot (\mathbf{c_1} \otimes \mathbf{c_2}) \right\rceil$. E_{prod} here denote the maximum noise in $\mathbf{c_{prod}}$ and $E_{prod} \leqslant \delta_1 + \delta_2 + \delta_3$, where $(\delta_i)_{1 \leqslant i \leqslant 3}$ is the noise inflicted by the key switching process. We refer the reader to [7, 8] for more detailed discussion on the properties of Brakerski's homomorphic cryptosystem.

3.4 Term Frequency-Inverse Document Frequency

One of the main problems of information retrieval is to determine the relevance of documents with respect to the user information needs. The most commonly used technique to represent the relevance score in the information retrieval community is Term Frequency-Inverse Documents Frequency measure [18, 27, 28]. It is computed based on two independent measures - the Term Frequency and the Inverse Document Frequency. The Term Frequency (TF) is a statistical measure that represents the frequency of repeated terms in a documents. The TF value calculates the number of times a given term appears within a single document. The Inverse Documents Frequency (IDF) is a measure of a term's importance across the whole document collection. It is defined as the logarithm of the ratio of the number of documents in a collection to the number of documents containing a given term.

We adopt the following equation for TF-IDF measure from [27]:

$$Score = \log\left(f_{i,j} + 1\right) \times \log\left(1 + \frac{n}{\sum_{k=1}^{n} \chi(f_{i,k})}\right) \tag{2}$$

where $f_{i,j}$ specifies the Term Frequency of term *j* in document D_i, *n* is the total number of documents in the corpus and $\sum_{k=1}^{n} \chi(f_{j,k})$ denotes IDF value for term *j* among the entire document collection D.

To provide the ranked results to user's queries we choose to use the dot product as similarity metric. We use vector space model [18], where the documents and search query are represented as high dimensional vectors. The similarity metric is measured by applying the dot product between each document vector and the search query vector as follows: $dotprod(D_i, Q) = D_i \otimes Q$, where D_i is a vector that represents *i*-th document and Q is a search query vector.

4 Multi-keyword Similarity Searchable Encryption (MKSim)

Recall that we are targeting the following scenario: the data owner creates secure searchable index and sends it along with encrypted data files to the cloud server. The index is constructed in such a way that it provides enough information to perform the search on the outsourced data, but does not give away any information about the original data. Once the server receives the index and encrypted document files, it performs a search on the index and retrieves the most relevant documents according to data user's query.

4.1 Algorithm Definitions

Definition 2. *Multi-keyword Similarity Searchable Encryption (MKSim) scheme over the set of documents* D *consists of five polynomial-time algorithms, as follows:*

1. $S_1, S_2, PK, SK \leftarrow Gen(1^s)$*: a probabilistic algorithm that is run by the data owner to setup the scheme. The algorithm outputs secret keys* S_1, S_2, PK *and* SK.
2. $(I, C) \leftarrow BuildIndex(S_1, S_2, PK, D, K)$*: a probabilistic algorithm run by the data owner that takes as input a collection of documents* D*, the keyword dictionary* K *and secret keys* S_1, S_2 *and* PK*, and outputs the collection of encrypted documents* $C = \{C_1, C_2, \ldots, C_n\}$ *and the searchable index* I.
3. $\Omega \leftarrow MakeQuery(S_2, PK, K, \bar{K})$*: a (possibly probabilistic) algorithm run by the data user that takes an input a secret keys* S_2 *and* PK*, keyword dictionary* K *and multiple keywords of interest* $\bar{K}$*, and outputs the search query* Ω.
4. $L \leftarrow Evaluate(I, \Omega)$*: a deterministic algorithm run by the cloud server. The algorithm inputs a search query* Ω *and the secure index* I*. It outputs the sequence of identifiers* $L \subseteq C$ *matching the search query.*
5. $D_i \leftarrow Decrypt(S_1, SK, C_i)$*: a deterministic algorithm that takes as input secret keys* S_1 *and* SK*, and a ciphertext* C_i *and outputs a document* D_i.

An index-based MKSim scheme is correct if $\forall\ s \in \mathbb{N}$, $\forall\ S_1, S_2, PK, SK$ *generated by* $Gen(1^s)$, $\forall\ D$, $\forall\ (I, C)$ *output by* $BuildIndex(S_1, S_2, PK, D, K)$, $\forall\ \bar{K} \in K$*, and* $1 \leqslant i \leqslant n$*,*

$$Evaluate(I, MakeQuery(S_2, PK, K, \bar{K})) = D(\bar{K}) \bigwedge Decrypt(S_1, SK, C_i) = D_i \tag{3}$$

4.2 MKSim Security Model

Definition 3. ***History:*** *Let* $\bar{K}$ *be a collection of keywords of interest,* D *be a collection of documents and* $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_q\}$ *be a vector of* q *search queries. The history of is defined as* $H(D, \Omega)$.

Definition 4. ***Access Pattern:*** *Let* $\bar{K}$ *be a collection of keywords of interest,* D *be a collection of documents and* $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_q\}$ *be a vector of* q *search queries. The access pattern induced by a* q*-query history* $H(D, \Omega)$*, is defined as follows:* $\alpha(H) = (D(\Omega_1), D(\Omega_2), \ldots, D(\Omega_q))$.

Definition 5. ***Search Pattern:*** *Let* $\bar{K}$ *be a collection of keywords of interest,* D *be a collection of documents and* $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_q\}$ *be a vector of* q *search queries, The search pattern of the history* $H(D, \Omega)$ *is a symmetric binary matrix* $\sigma(H)$ *such that for* $1 \leqslant i, j \leqslant q$*, the element in the* i^{th} *row and* j^{th} *column is 1 if* $\Omega_i = \Omega_j$*, and 0 otherwise.*

Definition 6. ***Trace:*** *Let* $\bar{K}$ *be a collection of keywords of interest,* D *be a collection of documents and* $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_q\}$ *be a vector of* q *search queries. The trace induced by the history* $H(D, \Omega)$ *is a sequence* $\tau(H) = (|D_1|, |D_2|, \ldots, |D_n|, \alpha(H), \sigma(H))$ *comprised of the lengths of document collection* $D = (D_1, D_2, \ldots, D_n)$*, and the access and search patters induced by the history* $H(D, \Omega)$.

Definition 7. ***Randomized query:*** *Let* $\Omega_{1\leqslant i\leqslant q}$ *be a sequence of* q *generated search queries with the same set of keywords* $\bar{K}$*. We say that the scheme has* (q, ϵ)*-randomized query if:* $\forall i, j \in 1, q \;\; Pr(\Omega_i = \Omega_j) < \epsilon$.

Definition 8. ***Adaptive Semantic Security****: Let* $MKSim = (Gen, BuildIndex, MakeQuery, Evaluate, Decrypt)$ *be an index-based similarity searchable encryption scheme,* s *be the security parameter, and* $A = (A_0, \ldots, A_q)$ *be an adversary such that* $q \in \mathbb{N}$ *and* $\mathbb{S} = (\mathbb{S}_0, \ldots, \mathbb{S}_q)$ *be a simulator. Consider the following probabilistic experiments* $Real^{\star}_{MKSim,A}(s)$ *and* $Sim^{\star}_{MKSim,A,S}(s)$*:*

$Real^{\star}_{MKsim,A}(s)$*:*	$Sim^{\star}_{MKSim,A,S}(s)$*:*
$(D, st_A) \leftarrow A_0(1^s)$	$(D, st_A) \leftarrow A_0(1^s)$
$S_1, S_2 \leftarrow$ *Gen*(1^s)	$(I, C, st_S) \leftarrow S_0(\tau(D))$
$(I, C) \leftarrow BuildIndex(S_1, S_2, D, K)$	$(\bar{K_1}, st_A) \leftarrow A_1(st_A, I, C)$
$(\bar{K_1}, st_A) \leftarrow A_1(st_A, I, C)$	$(\Omega_1, st_S) \leftarrow S_1(st_S, \tau(D, \bar{K_1}))$
$\Omega_1 \leftarrow MakeQuery(S_2, K, \bar{K_1})$	*for* $2 \leqslant i \leqslant q$
for $2 \leqslant i \leqslant q$	$(\bar{K_i}, st_A) \leftarrow A_i(st_A, I, C, \Omega_1, \ldots, \Omega_{i-1})$
$(\bar{K_i}, st_A) \leftarrow A_i(st_A, I, C, \Omega_1, \ldots, \Omega_{i-1})$	$(\Omega_i, st_S) \leftarrow S_i(st_S, \tau(D, \bar{K_1}, \ldots, \bar{K_i}))$
$\Omega_i \leftarrow MakeQuery(S_2, K, \bar{K_i})$	*let* $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_q\}$
let $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_q\}$	*output* $o = (I, C, \Omega)$ *and* st_A
output $o = (I, C, \Omega)$ *and* st_A	

where st_A *is the state of adversary,* st_S *is the state of the simulator. We say that MKSim is adaptively semantically secure if for all polynomial-size adversaries* $A = (A_0, \ldots, A_q)$ *such that* q *is is polynomial in* s*, there exist a non-uniform polynomial-size simulator* $\mathbb{S} = (\mathbb{S}_0, \ldots, \mathbb{S}_q)$ *such that for all polynomial-size* $\mathbb{R}$*:*

$$|Pr[\mathbb{R}(o, st_A) = 1] - Pr[\mathbb{R}(\bar{o}, \bar{st_A})]| \leqslant \epsilon, \tag{4}$$

where $(o, st_A) \leftarrow Real^{\star}_{MKSim,A}(s)$*,* $(\bar{o}, \bar{st_A}) \leftarrow Sim^{\star}_{MKSim,A,S}(s)$ *and the probabilities are over the coins of* Gen *and* $BuildIndex$ *and* $MakeQuery$*.*

4.3 MKSim Construction

Our searchable scheme is based on SSE-2 inverted index data construction previously introduced in [12]. We enhance SSE-2 scheme with addition of TF-IDF statistical measurement and dot product for ranked search. We show that our construction is very efficient and it achieves the same semantic security guarantees as SSE-2 scheme. Fig. 1 shows an outline of MKSim scheme.

Our searchable index consist of two main algorithms: building the lookup filter T, based on SSE-2 construction and building the TF-IDF table Φ, based on TF-IDF word importance. We first couple the document collection D with the dictionary K to produce the lookup filter T. For each word k we add an entry in T, there $value$ is the document identifier with the instance of word k. Note, for a given word k and the set of documents that contains the word k, we derive a label for k with j^{th} document identifier. For example, if word k is a "colorado" and there is only one document with this word,

then $k||j$ is "colorado1". We represent the family of k with matching j^{th} documents as follows: $F_k = \{k||j : 1 \leqslant j \leqslant |D(k)|\}$, where $|D(k)|$ represents the list of matching documents. For instance, if the word k="state" exists in a set of four documents, then family F_k is {"state1", "state2", "state3", "state4"}. In our construction, searching for word k becomes equal of searching for all labels in a form of $k||j$ in the family F_k.

We guard the unique number of words in each document by adopting the idea of padding the lookup filter T such that the identifier of each document appears in the same number of entries. To protect the keyword content in the table T, we use the pseudo-random permutation π with secret parameter S_2 such as $\{0,1\}^{S_2} \times \{0,1\}^{l+\log_2(n+max)} \rightarrow \{0,1\}^{l+\log_2(n+max)}$, where max denote the maximum number of distinct keywords in the largest document in D, n is the number of documents in D and each keyword is represented using at most l bits. Our lookup table T is $(\{0,1\}^{l+\log_2(n+max)} \times \{0,1\}^{\log_2(n)} \times \{p\})$, where $p = max \star n$.

In our second step, for each distinct keyword k_j in a document D_i, we calculate the TF-IDF value using equation (2). We then construct a table Φ where each row corresponds to the document id and each column is a keyword in the dictionary K. Each cell element of table Φ contains the TF-IDF value of a keyword k_j. Unfortunately, outsourcing the table elements to the cloud leaks some important information. It is well known fact [24] that an adversary (in our case, the cloud server) may know some of keywords and their TF distributions. Using this information, an adversary can infer the keyword index or even the document content. Based on this observation, we decided to improve the security of our solution. Our table includes the set of dummy keywords Z that are added to the keyword dictionary K. This gives us the randomness that hides the original keyword distribution of TF-IDF values. Finally, we use the "batching mode" encryption of Brakerski's homomorphic cryptosystem to protect the values of TF-IDF table Φ. We apply Brakerski's $Hom.Enc()$ with secret PK on each row of TF-IDF table Φ.

Once both the lookup filter T and TF-IDF table Φ are constructed, we use secure symmetric encryption scheme SKE to encrypt each document D_i with secret key S_1 to form C_i. We outsource searchable index $I = (T, \Phi)$ and encryption collection $C = \{C_1, C_2, \ldots, C_n\}$ to the cloud server.

Now the collection is available for selective retrieval from the cloud server. To search for keywords of interest $\bar{K}$, the data user uses the trapdoor to output the search query Ω to the cloud server. The trapdoor utilizes the pseudo-random permutation π with secret parameter S_2 and $Hom.Enc()$ with secret PK to form the search query Ω. The server then locates the document identifiers id in the filter table T that matches the keywords of interest. For each encrypted document C_j, where $1 \leqslant j \leqslant id$, the cloud server executes the dot product between the search query and the values in TF-IDF table Φ to form the set of polynomials $\{score_{C_1}, score_{C_2}, \ldots, score_{C_{id}}\}$. The data user decrypts these polynomials using $Hom.Dec()$ with secret SK and then he retrieves top-m documents with highest output scores.

$\underline{Gen(1^s)}$: generate $S_1 \leftarrow SKE.Gen(1^s)$ and $S_2 \xleftarrow{R} \{0,1\}^s$. Sample **SK**, **PK** $\leftarrow Hom.KeyGen()$. Output S_1, S_2, **SK** and **PK**.

$\underline{BuildIndex(S_1, S_2, PK, D, K)}$:

Initialization:

1. Scan document corpus D, extract $k_1, k_2, \ldots, k_p \leftarrow$ from D_i.
2. Construct dictionary K with dummy Z.
3. For each $k \in K$, build $D(k)$(i.e., the sequence of documents with k).

Build lookup filter T:

1. for each $k_i \in K$:
   - for $1 \leqslant j \leqslant |D(k_i)|$:
     - value = id($D_{i,j}$), where id($D_{i,j}$) is the j^{th} identifier in $D(k_i)$.
     - set $T[\pi_{S_2}(k_i||j)]$ = value.
2. let $\bar{p} = \sum_{k_i \in K} |D(k_i)|$. If $\bar{p} < p$, assign $value = id(D)$ for all $D_i \in D$ for exactly max entries, set the $address$ to random values.

Build TF-IDF table Φ:

1. for each $D_i \in D$
   - for each $k_j \in K$
     - $z_j \leftarrow$ TFIDF(k_j) (i.e., calculate TF-IDF value for keyword k_j).
   - $\boldsymbol{\Phi}_i = Hom.Enc(\mathbf{PK}, \mathbf{z})$.

Output:

1. for each $D_i \in D$, let $C_i \leftarrow SKE.Enc(S_1, D_i)$.
2. output (I, C), where $I = (T, \Phi)$ and $C = \{C_1, C_2, \ldots, C_n\}$.

$\underline{MakeQuery(S_2, PK, K, \bar{K})}$:

1. for each $k_i \in \bar{K} \in K$, $t_i = (\pi_{S_2}(k_i||1), \ldots, \pi_{2_1}(k_i||n))$, $b_i \leftarrow$ TFIDF(k_i).
2. $\mathbf{x} = Hom.Enc(\mathbf{PK}, \mathbf{b})$.
3. output: $\Omega = (t, \mathbf{x})$.

$\underline{Evaluate(I, \Omega)}$:

1. $id \leftarrow T[t]$.
2. for all $1 \leqslant j \leqslant id$, $score_{C_j} \leftarrow \boldsymbol{\Phi}_{C_j} \otimes \mathbf{x}$.
3. output $\{score_{C_1}, score_{C_2}, \ldots, score_{C_{id}}\}$.

$\underline{Decrypt(S_1, SK, C_i)}$:

1. $score_{D_i} = Hom.Dec(SK, score_{C_i})$.
2. $(score_{D_i})_m \xleftarrow{\text{top-m}} \{score_{D_1}, score_{D_2}, \ldots, score_{D_{id}}\}$.
3. output $(D_i)_m \leftarrow SKE.Dec(S_1, (C_i)_m)$.

Fig. 1. MKSim Construction

5 Security Analysis and Complexity Results

5.1 Security

Theorem 1. *If* $MKSim = (Gen, BuildIndex, MakeQuery, Evaluate)$ *is a index-based searchable encryption scheme, then it is adaptively semantically secure.*

Proof. We are going to describe a polynomial-size simulator $\mathbb{S} = \{\mathbb{S}_0, \mathbb{S}_1, \ldots, \mathbb{S}_q\}$ such that for all polynomial-size adversaries $A = \{A_1, A_2, \ldots, A_q\}$, the outputs of $Real^{\star}_{MKSim,A}(s)$ and $Sim^{\star}_{MKSim,A,S}(s)$ are computationally indistinguishable. Consider the simulator $\mathbb{S} = \{\mathbb{S}_0, \mathbb{S}_1, \ldots, \mathbb{S}_q\}$ that adaptively generates the output $o^{\star} = (I^*, C^*, \Omega^*)$ as follows:

- $\mathbb{S}_0(1^s, \tau(D))$: the simulator has a knowledge of history H that includes the number and the size of the documents. $\mathbb{S}_0$ start with generating $C_i^* \xleftarrow{R} \{0,1\}^{|D_i|}$ where $i \in \{1, 2, \ldots, n\}$ and index $I^* = (T^*, \Phi^*)$. Here $T^* \xleftarrow{R} \{0,1\}^{l+\log_2(n+max)}$ is a lookup

filter, and $\Phi^* \xleftarrow{R} \{0,1\}^K$ is TF-IDF table. $\mathbb{S}_0$ now includes I^* in st_S and outputs (I^*, C^*, st_S). Since st_A does not include secrets S_1 and **PK**, I^* is indistinguishable from the real index. Otherwise $\mathbb{S}_0$ can distinguish between the output of pseudo-random permutation π and a random values of size $l+\log_2(n+max)$ and K. At the same time, st_A does not include secret S_2, thus the output C^* is indistinguishable from the real ciphertext

- $\mathbb{S}_1(st_S, \tau(D, \Omega_1))$: now the simulator $\mathbb{S}_1$ has a knowledge of all document identifiers corresponding to the search query. However the search query does not disclose its structure and the content. Recall that $D(\Omega_1)$ is the set of all matching document identifiers. For all $1 \leqslant j \leqslant |D(\Omega_1)|$, the simulator first makes an association between each document identifier $id(D_j)$ and a generated search query such that $(D(\Omega_1)_i, I_i^\star)$ are pairwise distinct. $\mathbb{S}_1$ then creates $\Omega_1^\star = (t^\star, x^\star)$, where $t^\star \xleftarrow{R} (id(D_1), \ldots, id(|D(\Omega_1)|))$ and $x^\star \xleftarrow{R} \{0,1\}^K$. $\mathbb{S}_1$ stores the association between $\Omega_1^\star$ and Ω_1 in st_S, and outputs $(\Omega_1^\star, st_S)$. Since st_A does not include secret S_1, the output $t_1^\star$ is indistinguishable from the real generated query t_1, otherwise one could distinguish between the output of π and a random string of size $l+\log_2(n+max)$. Similarly, since st_A does not include secret **PK**, the output $x^\star$ is indistinguishable from the real x, otherwise one could distinguish between the output of $Hom.Enc()$ and a random string of a size K. Thus, $\Omega_1^\star$ is indistinguishable from Ω_1.
- $\mathbb{S}_i(st_S, \tau(D, \Omega_1, \Omega_2, \ldots, \Omega_q))$ for $2 \leqslant i \leqslant q$: first $\mathbb{S}_i$ checks if the search query Ω_i was executed before, that is, if it appeared in the trace $\sigma[i,j] = 1$, where $1 \leqslant j \leqslant i-1$. If $\sigma[i,j] = 0$, the search query has not appeared before and $\mathbb{S}_i$ generates the search query as $\mathbb{S}_1$. If $\sigma[i,j] = 1$, then $\mathbb{S}_i$ retrieves previously searched query, and constructs $\Omega_i^\star$. $\mathbb{S}_i$ outputs $(\Omega_i^\star, st_S)$, where $\Omega_i^\star$ is indistinguishable from real Ω_i. The final output $\Omega = (\Omega_1^\star, \ldots, \Omega_q^\star)$ is indistinguishable from generated query $(\Omega = \Omega_1, \ldots, \Omega_q)$, and outputs of experiments $Sim^\star_{MKSim,A,S}(s)$ and $Real^\star_{MKSim,A}(s)$ are indistinguishable.

□

Theorem 2. *Injection of dummy keywords provides randomized search queries.*

Proof. Let us consider two search queries Ω_1 and Ω_2, both constructed from the same keyword set K and a randomly chosen set of dummy keywords Z_1 and Z_2 from a dictionary $\mathscr{Z}$ of a size $n = |\mathscr{Z}|$. We are aiming to prove that: $\forall i, j \;\; i \neq j \;\; Pr(\Omega_i = \Omega_j) < \epsilon$ where ϵ tends to zero as n increases.

We first estimate the probability $Pr(k)$ that the intersection **Z** of two sets $Z_1, Z_2 \subseteq \mathscr{Z}$ is equal to some value k. We have:

$$Pr(k) = \frac{\#of\ ways\ of\ choosing\ Z_1, Z_2\ with\ |\mathbf{Z}| = k}{\#\ of\ ways\ of\ choosing\ Z_1, Z_2} \tag{5}$$

Note, there are $\binom{n}{k}$ choices for **Z**. If Z_1 has size k, then there is one choice for Z_1, and we can choose Z_2 arbitrarily from 2^{n-k} possibilities. If Z_1 has size $k+1$, then there are $\binom{n-k}{1}$ choices for Z_1 and 2^{n-k-1} choices for Z_2. Let $m = n-k$, then there are $\sum_{j=0}^{m} 2^{m-j}\binom{m}{j} = 3^m$ possible choices for Z_1, Z_2 with intersection **Z**. Thus, there are

$3^{n-k}\binom{n}{k}$ possible ways of choosing the subsets. Since there are 4^n ways of choosing any two subsets of $\mathscr{Z}$, we have the following: $Pr(k) = \frac{3^{n-k}\binom{n}{k}}{4^n}$. We now evaluate $Pr(k)$ with input $k \to 0$: $\lim_{k\to 0} Pr(k) = \lim_{k\to 0} \frac{3^{n-k}\binom{n}{k}}{4^n} = \left(\frac{3}{4}\right)^n$. As n increases, $\lim_{k\to 0} Pr(k) \to 0$ and hence Theorem 2 is preserved . □

5.2 Complexity

We compare MKSim scheme with previous searchable encryption solutions in Table 1. Our comparison is based on few simple metrics: matching technique, query randomization, security notions and search complexity. We use security notations from [12]. Note that all previous work able to achieve the linear search complexity within the total number of documents in the collection. In contrast, the search in our solution is proportional to the number of files that contain a certain set of keywords.

Table 1. Comparison of several searchable encryption schemes. n is the document collection, $|n|$ denotes its bit length, $\#n$ is the number of files in the collection n, $\#n_{\bar{K}}$ is the number of files that contain keywords of interest $\bar{K}$.

Scheme	Matching	Query randomization	Security	Search complexity
Song *et al.* [23]	Exact	no	CPA	O(|n|)
Goh *et al.* [16]	Exact	no	CKA1	O(#n)
Cao *et al.* [9]	Similarity	yes	CKA1	O(#n)
Moataz *et al.* [21]	Exact	yes	CKA2	O(#n)
Curtmola *et al.* [12]	Exact	no	CKA2	$O(\#n_{\bar{K}})$
Orencik *et al.* [22]	Exact	no	CKA2	O(#n)
MKSim	**Similarity**	**yes**	**CKA2**	**$O(\#n_{\bar{K}})$**

6 Conclusion

Searchable encryption is a technique that enables secure searches over encrypted data stored on remote servers. We define and solve the problem of multi-keyword ranked search over encrypted cloud data. In particular, we present an efficient similarity searchable encryption scheme that supports multi-keyword semantics. Our solution is based two building blocks: Term Frequency - Inverse Document Frequency (TF-IDF) measurement and ring-LWE-based variant of homomorphic cryptosystem. We use the dot product to quantitatively evaluate similarity measure and rank the outsourced documents with their importance to the search query. We show that our scheme is adaptive semantically secure against adversaries and able to achieve optimal sublinear search time. As future work, we plan to optimize the index construction algorithm and continue to research on usable and secure mechanisms for the effective utilization over outsourced cloud data.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: consistency properties, relation to anonymous ibe, and extensions. Journal of Cryptology 21(3), 350–391 (2008)
2. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order-preserving encryption for numeric data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France (June 2004)
3. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
4. Blanton, M.: Achieving full security in privacy-preserving data mining. In: Proceedings of the 3rd IEEE International Conference on Privacy, Security, Risk and Trust, Boston, MA, USA (October 2011)
5. Blanton, M., Atallah, M.J., Frikken, K.B., Malluhi, Q.: Secure and efficient outsourcing of sequence comparisons. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 505–522. Springer, Heidelberg (2012)
6. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Proceedings of the 4th IACR Theory of Cryptography Conference, Amsterdam, The Netherlands (February 2007)
7. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, Cambridge, MA, USA (2012)
9. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings of the 30th IEEE International Conference on Computer Communications, Shanghai, China (April 2011)
10. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
11. Di Crescenzo, G., Saraswat, V.: Public key encryption with searchable keywords based on jacobi symbols. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 282–296. Springer, Heidelberg (2007)
12. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, Virginia, USA (October 2006)
13. di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Private data indexes for selective access to outsourced data. In: Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, Chicago, IL, USA (October 2011)
14. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), `http://crypto.stanford.edu/craig`
15. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, Bethesda, MD (2009)
16. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003), `http://eprint.iacr.org/2003/216/`
17. Goldreich, O.: The foundations of cryptography. Basic Applications, vol. 2. Cambridge University Press (2004)

18. Grossman, D.A., Frieder, O.: Information retrieval: algorithms and heuristics. Kluwer international series on information retrieval. Springer (2004)
19. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman & Hall/CRC Cryptography and Network Security Series. Chapman & Hall/CRC (2007)
20. Kuzu, M., Islam, M.S., Kantarcioglu, M.: Efficient similarity search over encrypted data. In: Proceedings of the 28th IEEE International Conference on Data Engineering, Washington, DC, USA (April 2012)
21. Moataz, T., Shikfa, A.: Boolean symmetric searchable encryption. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, Hangzhou, China (May 2013)
22. Orencik, C., Kantarcioglu, M., Savas, E.: A practical and secure multi-keyword search method over encrypted cloud data. In: Proceedings of the 6th IEEE International Conference on Cloud Computing, Santa Clara, CA, USA (June 2013)
23. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy, Berkeley, CA, USA (May 2000)
24. Swaminathan, A., Mao, Y., Su, G.-M., Gou, H., Varna, A.L., He, S., Wu, M., Oard, D.W.: Confidentiality-preserving rank-ordered search. In: Proceedings of the ACM Workshop on Storage Security and Survivability, Alexandria, Virginia, USA (2007)
25. Wang, C., Cao, N., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Transactions on Parallel and Distributed Systems 23(8), 1467–1479 (2012)
26. Wang, C., Ren, K., Yu, S., Urs, K.M.R.: Achieving usable and privacy-assured similarity search over outsourced cloud data. In: Proceedings of the 31st Annual IEEE International Conference on Computer Communications, Orlando, FL, USA (March 2012)
27. Witten, I.H., Moffat, A., Bell, T.C.: Managing gigabytes: compressing and indexing documents and images, 2nd edn. Morgan Kaufmann, San Francisco (1999)
28. Zobel, J., Moffat, A.: Exploring the similarity space. SIGIR FORUM 32, 18–34 (1998)