Secure Multi-keyword Similarity Search Over Encrypted Cloud Data Supporting Efficient Multi-user Setup

Mikhail Strizhov[*], **Indrajit Ray**[*]

[*]Computer Science Department, Colorado State University, Fort Collins, CO, 80523, USA.

E-mail: `strizhov@CS.ColoState.EDU`, `Indrajit.Ray@ColoState.EDU`

Received 8 September 2015; received in revised form 22 January 2016; accepted 18 April 2016

Abstract. Searchable encryption allows one to store 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. This not only protects the data from the prying eyes of the server, but can also reduce the communication overhead between the server and the user and local processing at the latter. Previous research in searchable encryption have investigated exact match search on keywords and boolean expression search on keywords. In this work[1], we first propose a novel secure and efficient multi-keyword similarity searchable encryption (MKSim) that returns the matching data items in a ranked order manner. Unlike many existing schemes, our search complexity is sublinear to the total number of documents that contain the queried set of keywords. Our theoretical analysis demonstrates that the proposed scheme is provably secure against adaptive chosen-keyword attacks, the strongest form of security sought in searchable encryption. Next, we develop a proof-of-concept prototype that we use for experimentation on a large-scale real-world dataset and evaluate the efficiency and scalability of our solution. Finally, we extend the MKSim protocol to the multi-user setting in which the data owner wishes to provide selective access to his encrypted document corpus to more than one user.

Keywords. searchable encryption, multi-keyword ranked search, homomorphic encryption, distributed broadcast 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 and management. 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,

[1]The current work is the extended version of [14].

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, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19]. Solutions generally involve 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 in terms of the types of techniques they use to build the trapdoor function that facilitates the search. A few of them, most notably [4, 5, 6], 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 resistant against *adaptive* adversaries [9]. Taking into account large volumes of data available today, there is need for efficient methods to perform secure similarity search over encrypted data outsourced into the cloud. Finally, these works are mostly confined to the single user setting – the owner of the encrypted document corpus is the one to search it. If an arbitrary group of data users need to search the encrypted document corpus, existing schemes fail to manage their access privileges. 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. In this work, we focus on the problem of constructing similarity searchable encryption scheme for the purpose of making *practical* searchable cryptographic cloud storage systems.

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[20, 9]. 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 many 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 perform a thorough experimental evaluation of our solution on a real-world dataset.

4. We extend the searchable encryption to multi-user setting, where an arbitrary group of data users can submit queries to the remote server to search an encrypted document collection. We design a scheme supports distributed setup, where participants choose their own secret key rather than receive the key from a trusted authority.

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. Security analysis is given in Section 5. Performance evaluation as well as comparison to other existing schemes is given in Section 6. The extension of our solution towards an arbitrary group of users is presented in Section 7. Finally, Section 8 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, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 18, 15, 16, 19, 17, 21, 22]. 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 search. Song *et al.* [1] were the first to investigate the techniques for keyword search over encrypted outsourced data. The authors begin with the idea to store a set of plaintext documents on data storage servers (e.g., 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 [2] developed a Bloom filter-based secure index, which results in the search cost proportional to the number of files in collection. The scheme requires a linear search time and provides some false positive results. Curtmola *et al.* [9] gave the first solution that enables sublinear search time over the encrypted document collection. In proposed solution the searchable index consists of a keyword trapdoor and encrypted document identifiers whose corresponding data files contain the keyword. Moreover, they proposed a multi-user solution that includes an arbitrary group of users (authorized by the data owner) that can submit search queries to the cloud server. Moataz *et al.* [15] proposed a boolean symmetric searchable encryption scheme. Here, the scheme is based on the orthogonalization of keywords according to the Gram-Schmidt process. Orencik's solution [16] proposed a privacy-preserving multi-keyword search method that utilizes minhash functions. Recently, Strizhov *et al.*[28] presented a substring searchable encryption scheme that allows substring search queries over encrypted cloud data and also finds positions of the substring within encrypted document collection.

Boneh *et al.* [3] developed the first searchable encryption using an asymmetric key settings, where anyone with the public key can write to the data stored remotely, but the users with private key invoke search queries. Another asymmetric solution is described by Di Crescenzo *et al.* in [21], 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 usability since they support only *exact* match in the context of keyword search without considering the frequency/importance of keywords in the document collection. Wang *et al.* [4] studied the problem of secure ranked keyword search over encrypted cloud data. The authors explore a 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 most relevant documents. Cao *et al.* [5] present a multi-keyword ranked search scheme, where they used the principle of "coordinate matching" that captures the similarity between a search query (that includes multiple keywords) and encrypted documents. Here, the search query and stored documents are represented as high dimensional vectors and similarity measure is used to determine the rank of each document. However, their index structure uses a binary representation of document terms, and thus the ranked search does not differentiate between documents with higher number of repeated terms from documents with lower number of repeated terms (e.g., documents which contain all query keywords are ranked equally).

The problem of searching on encrypted data can be solved using the work of Goldreich *et al.* [23] on Oblivious RAM (ORAM). This approach provides the strongest levels of security, where the remote server does not learn any information about the stored data or the search queries (including the access pattern). However, this approach requires many rounds of interaction and has a high overhead for the remote server and the client. Stefanov *et al.* [25] propose a relatively practical ORAM scheme; however it involves large communication and storage costs. Moataz *et al.* [26] develop a tree-based ORAM scheme that aims to achieve a sublinear communication complexity. However, recent work by Naveed *et al.* [27] showed that using ORAM in searchable encryption does not completely eliminate the access pattern leakage and does not provide any meaningful reduction in leakage.

Blanton *et al.* [6, 8] look at the problem of secure outsourcing of sequence comparisons to remote servers. Their approach uses the edit distance [24] metric to measure the similarity between two strings with different lengths. This technique can be utilized for similarity search over encrypted data. However, the technique is quite inefficient owing to the high computational overhead involving a large number of comparisons.

3 Background and Building Blocks

3.1 System & 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 $D = (D_1, D_2, \ldots, D_n)$ in encrypted form $C = (C_1, C_2, \ldots, C_n)$ to the cloud server and still preserve the search functionality on outsourced data. $C_i = E_S[D_i]$ 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 $K = (k_1, k_2, \ldots, k_m)$ 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 $\bar{K} \in K$, 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 $L \subseteq C$ 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* threat 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 element 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 $x \geq 0$)to indicate rounding down or up.

Definition 1. Symmetric Key Encryption Scheme (SKE): A symmetric encryption scheme $SKE = (Gen, Enc, Dec)$ consists of three algorithms, as follows:

- Gen: the *key generation algorithm*, is a probabilistic algorithm that returns a string K. Let $Keys(SKE)$ denote the set of all strings that have non-zero probability of being output by Gen. The members of this set are called keys. We denote $K \leftarrow Gen$ for the operation of executing Gen and letting K denote the key returned.
- Enc: the *encryption algorithm*, is a probabilistic algorithm that takes a key $K \in Keys(SKE)$ and a plaintext $M \in \{0,1\}^\star$. It returns a ciphertext $C \in \{0,1\}^\star$. We write $C \leftarrow Enc(K, M)$ for the operation of executing Enc on K and M and letting C denote the ciphertext returned.
- Dec: the *decryption algorithm*, a deterministic algorithm that takes a key $K \in Keys(SKE)$ and a ciphertext $C \in \{0,1\}^\star$. It returns some $M \in \{0,1\}^\star$. We write $M \leftarrow Dec(K, C)$ for the operation of executing Dec on K and C and letting M denote the message returned.

The SKE scheme is correct if for any key $K \in Keys(SKE)$, any sequence of messages (M_1, ..., M_Q) $\in \{0,1\}^\star$, and any sequence of ciphertexts ($C_1 \leftarrow Enc(K, M_1)$, ..., $C_Q \leftarrow Enc(K, M_Q)$) that may arise in encrypting (M_1, ..., M_Q), it is the case that $Dec(K, C_i)$ $= M_i$ for each $C_i \neq \perp$ ($i \in [1; Q]$). A symmetric-key encryption scheme SKE is secure against chosen-plaintext attacks (CPA) (executed by the adversary with an access to encryption oracle) if produced ciphertexts do not leak any useful information about the original plaintexts[7].

We also make use of pseudo-random function (PRF), which is a polynomial-time computable function that cannot be distinguished from random functions by any probabilistic polynomial-time adversary.

Definition 2. Pseudorandom function (PRF): A function $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^l$ is a (t, ϵ, q)-pseudorandom function if

- Given a key $K \in \{0,1\}^k$ and an input $X \in \{0,1\}^n$ there is an algorithm to compute $F_K(X) = F(X, K)$.
- For any t-time oracle algorithm A, we have:

$$|Pr_{K \leftarrow \{0,1\}^k}[A^{f_K}] - Pr_{f \in \mathrm{F}}[A^f]| < \epsilon \tag{1}$$

where $\mathrm{F} = \{f : \{0,1\}^n \to \{0,1\}^l\}$ and A makes at most q queries to the oracle.

3.3 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[31, 32, 33]. 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 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 given collection to the number of documents containing a given term. The consequence of IDF definition is that rarely occurring terms have high IDF values and common terms have low IDF value. In judging the value of a term for ranking representation, two different statistical criteria come into consideration. A term appearing often in one single document is assumed to carry more importance for content representation than a rarely occurring term. On the other hand, if that same term occurs as frequently in other documents of the collection, the term is possibly not as valuable as some other terms that occur less frequently in the remaining documents. An example of this conundrum is illustrated by the occurrence of prepositions in English language documents. This suggests that the ranking of a given term as applied to a given document can be measured by a combination of its frequency of occurrence and an inverse function of the number of documents in the collection. Therefore, we use the product of term frequency and the inverse documents frequency (TF-IDF) for the ranking function in our search. A term with higher TF-IDF value is more relevant to the user's query than the term with lower TF-IDF value.

We adopt the following equation for TF-IDF measure from [33]:

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

where $f_{i,j}$ specifies the TF value of term j in the document D_i, *n* is the total number of documents in the corpus and $\frac{n}{\sum\limits_{k=1}^{n} \chi(f_{i,k})}$ denotes the IDF value of term *j* among the entire 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[31], 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:

$$dotprod(D_i, Q) = D_i \otimes Q, \tag{3}$$

where D_i is a vector that represents *i*-th document and Q is a query vector.

3.4 Homomorphic Cryptosystem

We now review definitions related to homomorphic cryptosystem. Our definitions are based on Gentry's works [29] and [30], but we slightly relax the definition of decryption correctness, to allow a negligible probability of error.

Definition 3. Homomorphic encryption (Hom): Let s denote the security parameter. Fix a function $l = l(s)$. An l-homomorphic encryption Hom for a class of circuits $\{\mathbb{C}_s\}_{s\in\mathbb{N}}$ consists of four polynomial-time algorithms Gen, Enc, Dec, and $Eval$ such that:

- Gen: The *key generation algorithm*, is a probabilistic algorithm that takes the security parameter s as input and outputs a public key PK and secret key SK.
- Enc: The *encryption algorithm*, is a probabilistic algorithm that takes a public key PK and a message $m \in \{0, 1\}$ as input, and outputs a ciphertext c.
- Dec: The *decryption algorithm*, is a deterministic algorithm that takes the secret key SK and a ciphertext c as input, and outputs a message $m \in \{0, 1\}$.
- $Eval$: The *homomorphic evaluation algorithm*, takes as input a public key PK a circuit $C \in \mathbb{C}_s$, and a list of ciphertexts $c_1, \ldots, c_{l(s)}$, and outputs a ciphertext $c^\star$.

The following correctness properties are required to hold:

1. For any s, any $m \in \{0, 1\}$, and any (PK, SK) output by $Gen(s)$, we have $m = Dec(SK, Enc(PK, m))$.
2. For any s, any $m, \ldots, m_l$, and any $C \in \mathbb{C}_s$, we have

$$C(m_1, \ldots, m_l) = Dec(SK, Eval(PK, (C, Enc(PK, m), \ldots, Enc(PK, m_l)))) \quad (4)$$

We use the standard notion of security against chosen-plaintext attacks (CPA-security).

Definition 4. CPA security: Let $Hom = Gen, Enc, Dec, Eval$ be a homomorphic encryption scheme. Let s be the security parameter, A be an adversary and there is a probabilistic experiment $CPA_{Hom,A}(s)$ that is executed between the challenger and the adversary:

- Generate $(PK, SK) \leftarrow Gen(s)$.
- The adversary outputs $(m_0, m_1, st_A) \leftarrow A_1^{(\cdot)}(PK)$.
- The bit is chosen at random, i.e $b \leftarrow \{0, 1\}$.
- The adversary runs number of polynomial queries $c \leftarrow Enc(PK, m_b)$.
- The adversary outputs bit $b^{'} \leftarrow A_2^{(\cdot)}(c, st_A)$.

Homomorphic encryption scheme Hom is CPA-secure if for all polynomial-size adversaries A,

$$Pr[CPA_{Hom,A}(s) = 1] \leq \frac{1}{2} + negl(s), \quad (5)$$

where the probability is over the choice of bit b and the coins of Gen and Enc.

There are many homomorphic encryption scheme available in the literature. However, in this paper, we use Brakerski *et al.*[34, 35] homomorphic cryptosystem since it provides the efficient "batching mode" property 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.

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 5. Multi-keyword Similarity Searchable Encryption (MKSim): An index-based MKSim scheme over a set of documents D is a tuple of five polynomial-time algorithms $(Gen, BuildIndex, MakeQuery, Evaluate, Decrypt)$, 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 a set of secret keys S_1, S_2, SK and public key PK.

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, a keyword dictionary K and keys S_1, S_2 and PK, and outputs a collection of encrypted documents $C = \{C_1, C_2, \ldots, C_n\}$ and searchable index I.

3. $\Omega \leftarrow MakeQuery(S_2, PK, K, \bar{K})$: a probabilistic algorithm run by the data user that inputs keys S_2 and PK, keyword dictionary K and multiple keywords of interest $\bar{K}$, and outputs the search query Ω.

4. $L \leftarrow Evaluate(PK, I, \Omega, C)$: a deterministic algorithm run by the cloud server. The algorithm inputs a public key PK, searchable index I, search query Ω and set of encrypted documents C. The algorithm outputs a sequence of identifiers $L \subseteq C$ matching the search query.

5. $D_i \leftarrow Decrypt(S_1, SK, L_i)$: 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 $1 \leq i \leq n$,

$$Evaluate(PK, I, MakeQuery(S_2, PK, K, \bar{K}), C) = L \bigwedge Decrypt(S_1, SK, L_i) = D_i \quad (6)$$

4.2 MKSim Security Model

In this section we focus on security definitions for our scheme. Security goal of any searchable encryption scheme is to reveal nothing (no meaningful information) to the adversary. Our goal is to provide two following security guarantees:

- Given a searchable index I and a set of encrypted document $C = (C_1, C_2, \ldots, C_n)$ to the adversary, no valuable information about the original documents $D = (D_1, D_2, \ldots, D_n)$ is leaked to the adversary.

- Given a set of search queries $Q = (Q_1, Q_2, \ldots, Q_m)$ generated by the data user, the adversary cannot learn any information about the content of the search query Q_i or the content of the documents in the search outcome.

To hide the plaintext document collection we require the symmetric key encryption scheme SKE (see Definition 1) to have the pseudo-randomness against chosen-plaintext attacks (PCPA) security guarantee which assures that the ciphertexts are indistinguishable from random[2]. We outline the PCPA-security of SKE scheme as following experiment:

Definition 6. PCPA security: Let $SKE = (Gen, Enc, Dec)$ be a symmetric key encryption scheme, s be the security parameter, A be an adversary and there is a probabilistic experiment $PCPA_{SKE,A}(s)$ that is run as follows:

- Output the secret key $S_1 \leftarrow Gen(1^s)$.
- The adversary A is given oracle access to $Enc(S_1, \cdot)$.
- The adversary A outputs a message M (e.g., plaintext document D).
- Let $C_0 \leftarrow Enc(S_1, M)$ and $C_1 \overset{R}{\leftarrow} \mathbb{C}$, where $\mathbb{C}$ denotes the set of all possible ciphertexts. A bit b is chosen at random and C_b is given to the adversary A.
- The adversary A is again given to the oracle access to $Enc(S_1, \cdot)$, and A runs number of polynomial queries to output a bit $b^{'}$.
- The experiment outputs 1 if $b = b^{'}$, otherwise 0.

Symmetric key encryption scheme SKE is PCPA-secure if for all polynomial-size adversaries A,

$$Pr[PCPA_{SKE,A}(s) = 1] \leq \frac{1}{2} + negl(s), \tag{7}$$

where the probability is over the choice of bit b and the coins of Gen and Enc.

Achieving the second security property is difficult and and most known searchable encryption solutions [15, 37, 9, 2, 17] reveal some information from the search queries, namely *access pattern* and *search pattern*. In MKSim scheme we follow the similar approach to weaken the security guarantees and allow some limited information to the adversary. We begin with definitions of the *history*, *access pattern* and *search pattern*, and then we give the definition of *trace* that combines definitions of history, access and search patterns.

Definition 7. 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 search queries is defined as $H(D, \Omega)$.

Definition 8. 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: $\alpha(H) = (D(\Omega_1), D(\Omega_2), \ldots, D(\Omega_q))$, where $D(\Omega_i)$ denotes the set of documents that match search query Ω_i.

[2]Note that symmetric key encryption schemes such as AES in counter mode satisfy the PCPA-security definition.

Definition 9. 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 $1 \leq i, j \leq q$, the element in the i^{th} row and j^{th} column is 1 if $\Omega_i = \Omega_j$, and 0 otherwise.

The access pattern represents the results of search queries Q sent to the cloud server, and specifically, the document identifiers corresponding to each query. The search pattern represents a historical observation of queries searched for. We combine both access pattern and search pattern to form the definition of *trace*, as follows:

Definition 10. 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 patterns induced by the history $H(D, \Omega)$.

Unlike security definitions originally presented in [9], our searchable encryption scheme introduces a randomization of a search query that enables the data user to hide the search pattern. Thus, queries are different even if they are generated for the same set of keywords of interest. The definition of randomized queries is given as:

Definition 11. Randomized query: Let $\Omega_{1 \leq i \leq q}$ 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: $\forall i, j \in 1, q \;\; Pr(\Omega_i = \Omega_j) < \epsilon$.

Our security model uses the simulator-based definition approach that includes the view from adversary and the simulator. Adaptive security means that the adversary generates the history adaptively, that means that the results of previous search queries are taken into account. We outline simulation-based experiments in the following definition:

Definition 12. 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 $Real^{\star}_{MKSim,A}(s)$ that is executed between the challenger and the adversary, and $Sim^{\star}_{MKSim,A,\mathbb{S}}(s)$ that is executed between the adversary and the simulator:

$Real^{\star}_{MKsim,A}(s)$:	$Sim^{\star}_{MKSim,A,\mathbb{S}}(s)$:
$(D, K, st_A) \leftarrow A_0(1^s)$ $(S_1, S_2, PK, SK) \leftarrow Gen(1^s)$ $(I, C) \leftarrow BuildIndex(S_1, S_2, PK, D, K)$ $(\bar{K}_1, st_A) \leftarrow A_1(st_A, I, C)$ $\Omega_1 \leftarrow MakeQuery(S_2, PK, K, \bar{K}_1)$ for $2 \leq i \leq q$ $(\bar{K}_i, st_A) \leftarrow A_i(st_A, I, C, \Omega_1, \ldots, \Omega_{i-1})$ $\Omega_i \leftarrow MakeQuery(S_2, PK, K, \bar{K}_i)$ let $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_q\}$ output $o = (I, C, \Omega)$ and st_A	$(D, K, st_A) \leftarrow A_0(1^s)$ $(I, C, st_{\mathbb{S}}) \leftarrow \mathbb{S}_0(\tau(D, K))$ $(\bar{K}_1, st_A) \leftarrow A_1(st_A, I, C)$ $(\Omega_1, st_{\mathbb{S}}) \leftarrow \mathbb{S}_1(st_{\mathbb{S}}, \tau(D, \bar{K}_1))$ for $2 \leq i \leq q$ $(\bar{K}_i, st_A) \leftarrow A_i(st_A, I, C, \Omega_1, \ldots, \Omega_{i-1})$ $(\Omega_i, st_{\mathbb{S}}) \leftarrow \mathbb{S}_i(st_{\mathbb{S}}, \tau(D, \bar{K}_1, \ldots, \bar{K}_i))$ let $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_q\}$ output $o = (I, C, \Omega)$ and st_A

where st_A is the state of adversary, $st_{\mathbb{S}}$ 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 polynomial in s, there exists a non-uniform polynomial-size simulator $\mathbb{S} = (\mathbb{S}_0, \ldots, \mathbb{S}_q)$ such that for all polynomial-size $\mathbb{R}$:

$$|Pr[\mathbb{R}(o, st_A) = 1] - Pr[\mathbb{R}(\bar{o}, \bar{st_A})]| \leq \epsilon, \tag{8}$$

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

This completes our security model requirement for MKSim solution, and we are now ready to present scheme details.

4.3 MKSim Construction

Setup Phase. Our searchable scheme is based on SSE-2 inverted index data construction previously introduced in [9]. 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 consists 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 create a lookup filter T whose entries have of the form $(keyword, value)$. For each keyword $k \in K$ we add an entry in T whose $value$ is the document identifier with the instance of keyword k. Note, for a given keyword k and the set of documents that contains the keyword k, we derive a label $k||j$ for keyword k with j-th document identifier. For example, if "colorado" is contained in document D_5, then $k||j$ is "colorado1". Formally, we represent the family of k with matching j-th documents as follows: $F_k = \{k||j : 1 \leq j \leq |D(k)|\}$, where $|D(k)|$ represents the list of matching documents. For instance, if "state" is contained in set of four document $(D_2, D_4, D_{10}, D_{13})$, then family F_k is {"state1", "state2", "state3", "state4"} and we add the following entries in lookup table T: $(state1, 2)$, $(state2, 4)$, $(state3, 10)$ and $(state4, 13)$. In our construction, searching for keyword k becomes equal to 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 measure using Equation 2. We then construct the TF-IDF table Φ where each row corresponds to the document identifier D_i and each column is the keyword in the dictionary K. Each cell element in 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[38] 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 CPA-secure homomorphic cryptosystem

```
$Gen(1^s)$ : generate $S_1 \leftarrow SKE.Gen(1^s)$ and $S_2 \overset{R}{\leftarrow} \{0,1\}^s$. Sample $SK$, $PK \leftarrow$
$Hom.KeyGen()$. Output $S_1$, $S_2$, $SK$ and $PK$.
$BuildIndex(S_1, S_2, PK, D, K)$ :
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 $1 \leq j \leq |D(k_i)|$:
           – 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 $\Phi$:
  1. for each $D_i \in D$
       • for each $k_j \in K$
           – $WT_{i,j} \leftarrow$ TFIDF($k_j$) (i.e., calculate TF-IDF value for keyword $k_j$).
       • set $\mathbf{\Phi_i} = Hom.Enc(PK, \mathbf{WT_i})$ (i.e., apply homomorphic encryption $Hom$ with
         public key $PK$ on vector $WT_i$).
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 $C = \{C_1, C_2, \ldots, C_n\}$.
$MakeQuery(S_2, PK, K, \bar{K})$ :
  1. for each keyword $k_j \in \bar{K}$, set $t_j = (\pi_{S_2}(k_j||1), \ldots, \pi_{S_1}(k_j||n))$.
  2. for each keyword $k_j \in \bar{K}$, calculate $WT_j \leftarrow$ TFIDF($k_j$).
  3. set $\mathbf{x} = Hom.Enc(PK, \mathbf{WT})$ (i.e., encrypt vector $\mathbf{WT}$ using homomorphic encryption
     $Hom$ with public key $PK$).
  4. output search query $\Omega = (t, \mathbf{x})$.
$Evaluate(PK, I, \Omega, C)$ :
  1. Find all matching identifiers $id \leftarrow T[t]$.
  2. for all $j \in [1; |id|]$, calculate $score_{C_j} = Hom.Eval(PK, (\mathbf{\Phi}_{C_j}, \mathbf{x}))$. (i.e. with public key
     $PK$, execute homomorphic dot product (multiplication and addition) between $\mathbf{\Phi_{C_j}}$
     and $\mathbf{x}$).
  3. output results $\{score_{C_1}, \ldots, score_{C_{id}}\}$.
$Decrypt(S_1, SK, L_i)$ :
  1. set $score_{D_i} = Hom.Dec(SK, score_{L_i})$ (i.e., decrypt $score_{C_i}$ using $Hom$ with secret key
     $SK$).
  2. select $(score_{D_i})_m \xleftarrow{\text{top-m}} \{score_{D_1}, \ldots, score_{D_{id}}\}$ (i.e. select top-$m$ documents with
     highest scores).
  3. output $(D_i)_m \leftarrow SKE.Dec(S_1, C_i)$, where $i \in [1; m]$.
```

Figure 1: MKSim Construction

Hom to protect the values of TF-IDF table Φ. We apply $Hom.Enc()$ with public key PK on each row of TF-IDF table Φ.

Once both the lookup filter T and TF-IDF table Φ are constructed, we use PCPA-secure symmetric key encryption scheme SKE with secret key S_1 to encrypt each document D_i, i.e. $C_i = SKE.Enc(S_1, D_i)$. We outsource searchable index $I = (T, \Phi)$ and encrypted 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.

Search Phase. To search the keywords of interest $\bar{K}$, the data user contacts the data owner to receive the *search trapdoor*. The trapdoor includes the keyword dictionary K with IDF values for each keyword in K, the set of (S_1, S_2, PK, SK) keys, the pseudo-random permutation π, homomorphic cryptosystem Hom and symmetric-key encryption scheme SKE. Note, the trapdoor learning process is a one-time operation and the data user does not need to contact the data owner anymore. The data owner inputs the set of keywords of interest $\bar{K}$ to the search trapdoor. The trapdoor generates a search query in form of $\Omega = (t, x)$, where t corresponds to the lookup search query and x corresponds to the TF-IDF search query. The lookup search query t is the output of the pseudo-random permutation π with secret key S_2. The TF-IDF search query x is the result of applying the homomorphic encryption $Hom.Enc$ with public key PK on the output of TF-IDF measure calculated using Equation 2 on set of keywords of interest $\bar{K}$. Finally, the data user sends resulted search query $\Omega = (t, x)$ to the cloud server.

Once Ω received, the server locates the matching document identifiers in the lookup filter T as $T[t]$. Now the cloud server executes the homomorphic dot product $Hom.Eval$ (that includes homomorphic multiplication and homomorphic addition) between the TF-IDF search query x and rows in the TF-IDF table Φ that correspond to the matching document identifiers. The cloud server sends the set of resulted ciphertexts $\{score_{C_1}, \ldots, score_{C_{id}}\}$ back to the data user.

The data user decrypts each polynomial $score_{C_i}$ using $Hom.Dec$ with secret key SK to form $score_{D_i}$. The data user retrieves the top-m documents with highest similarity scores from the cloud server. Using $SKE.Dec$ with secret key S_1, the data user decrypts matching documents to the original view.

5 Security Analysis

We now focus on the security evaluation of proposed searchable encryption scheme. We focus on two security properties:

- We prove that our scheme is CKA2 secure[20, 9] and achieves the strongest semantic security against adaptive adversaries.
- We demonstrate that inserting dummy keywords provides randomized search queries. It is important to show that the search queries generated by our scheme are different for the same set of keywords of interest, making the adversary difficult to collect and distinguish the document outcomes.

Theorem 13. *If π is a pseudo-random permutation, Hom is CPA-secure and SKE is PCPA-secure, then $MKSim = (Gen, BuildIndex, MakeQuery, Evaluate)$ scheme is adaptively secure.*

Proof. We are going to describe a polynomial-size simulator $\mathbb{S} = \{\mathbb{S}_0, \ldots, \mathbb{S}_q\}$ such that for all polynomial-size adversaries $A = \{A_0, \ldots, A_q\}$, the outputs of $Real^{\star}_{MKSim,A}(s)$ and

$Sim^{\star}_{MKSim,A,\mathbb{S}}(s)$ are computationally indistinguishable. Consider the simulator $\mathbb{S} = \{\mathbb{S}_0, \ldots, \mathbb{S}_q\}$ that adaptively generates the output $o^{\star} = (I^{\star}, C^{\star}, \Omega^{\star})$ 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. $\mathbb{S}_0$ starts with generating $C^{\star}_i \xleftarrow{R} \{0,1\}^{|D_i|}$ where $i \in [1,n]$ and searchable index $I^{\star} = (T^{\star}, \Phi^{\star})$. Here $T^{\star} \xleftarrow{R} \{0,1\}^{l+\log_2(n+max)}$ is a lookup filter, and $\Phi^{\star} \xleftarrow{R} \{0,1\}^{K\times n}$ is a TF-IDF table. $\mathbb{S}_0$ now includes $I^{\star}$ in $st_{\mathbb{S}}$ and outputs $(I^{\star}, C^{\star}, st_{\mathbb{S}})$. Since, with all but negligible probability, st_A does not include secret S_2, $T^{\star}$ is indistinguishable from the real lookup table T. Otherwise $\mathbb{S}_0$ can distinguish between the output of pseudo-random permutation π and a random values of size $l + \log_2(n + max)$. Similarly, simulated TF-IDF table $\Phi^{\star}$ is indistinguishable from the real Φ due to the CPA security of homomorphic encryption Hom, otherwise one could distinguish between the output of $Hom.Enc$ and a random string of size $K \times n$. At the same time, st_A does not include secret S_1, thus the PCPA security of SKE scheme will guarantee that the output $C^{\star}$ is indistinguishable from the real ciphertext.

- $\mathbb{S}_1(st_{\mathbb{S}}, \tau(D, \Omega_1))$: now the simulator $\mathbb{S}_1$ has a knowledge of all document identifiers corresponding to the search query Ω_1. 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 $1 \leq j \leq |D(\Omega_1)|$, the simulator first makes an association between each document identifier $id(D_j)$ and a generated search query such that $(D(\Omega_1)_i, I^{\star}_i)$ are pairwise distinct. $\mathbb{S}_1$ then creates $\Omega^{\star}_1 = (t^{\star}, x^{\star})$, 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 $\Omega^{\star}_1$ and Ω_1 in $st_{\mathbb{S}}$, and outputs $(\Omega^{\star}_1, st_{\mathbb{S}})$. Since, with all but negligible probability, st_A does not include secret S_2, the output $t^{\star}_1$ 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, simulated $x^{\star}$ is indistinguishable from the real x due to the CPA security of homomorphic encryption Hom, otherwise one could distinguish between the output of $Hom.Enc()$ and a random string of a size K. Thus, $\Omega^{\star}_1$ is indistinguishable from Ω_1.

- $\mathbb{S}_i(st_{\mathbb{S}}, \tau(D, \Omega_1, \Omega_2, \ldots, \Omega_q))$ for $2 \leq i \leq q$: 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 $1 \leq j \leq i-1$. 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 $\Omega^{\star}_i$. $\mathbb{S}_i$ outputs $(\Omega^{\star}_i, st_{\mathbb{S}})$, where $\Omega^{\star}_i$ is indistinguishable from real Ω_i. The final output $\Omega = (\Omega^{\star}_1, \ldots, \Omega^{\star}_q)$ is indistinguishable from generated query $(\Omega = \Omega_1, \ldots, \Omega_q)$, and outputs of experiments $Sim^{\star}_{MKSim,A,\mathbb{S}}(s)$ and $Real^{\star}_{MKSim,A}(s)$ are indistinguishable.

□

Theorem 14. *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 $\mathcal{Z}$ of a size $n = |\mathcal{Z}|$. 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 $\mathrm{Pr(k)}$ that the intersection $\mathbf{Z}$ of two sets $\mathrm{Z}_1, \mathrm{Z}_2 \subseteq \mathcal{Z}$ is equal to some value k. We have:

$$Pr(k) = \frac{\#of\ ways\ of\ choosing\ Z_1, Z_2\ with\ |\mathbf{Z}| = k}{\#\ of\ ways\ of\ choosing\ Z_1, Z_2} \tag{9}$$

Note, there are $\binom{n}{k}$ choices for $\mathbf{Z}$. 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 $\mathrm{k}+1$, then there are $\binom{n-k}{1}$ choices for Z_1 and 2^{n-k-1} choices for Z_2. Let $\mathrm{m} = \mathrm{n} - \mathrm{k}$, then there are $\sum\limits_{j=0}^{m} 2^{m-j}\binom{m}{j} = 3^m$ possible choices for $\mathrm{Z}_1, \mathrm{Z}_2$ with intersection $\mathbf{Z}$. 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 $\mathcal{Z}$, we have the following: $Pr(k) = \frac{3^{n-k}\binom{n}{k}}{4^n}$. We now evaluate $\mathrm{Pr(k)}$ 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 14 is preserved . □

6 Performance Evaluation

In this section, we will focus on the performance evaluation of proposed solution. We first present the complexity analysis of $MKSim$ scheme and then focus on thorough experimental evaluation of our solution using a real-world dataset.

6.1 Complexity Analysis

We compare $MKSim$ scheme with previous searchable encryption solutions in Table 1. Our comparison is based on the following metrics: matching technique, query randomization, security, search time and index size. Matching technique describes if solution supports *exact* match or *similarity* match. Query randomization describes the support of randomized search queries. We use security notations from [9] to describe the security of each solution. Search time is the time to invoke search on the index that is constructed from the document collection D of size n. Note that previous solutions are 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 documents that contain a set of keywords of interest. Finally, we measure the size of the searchable index. Our searchable index I consists of a lookup filter T of size $O(nm)$ (where n is the size of document collection and m is the size of keyword dictionary K) and a TF-IDF table Φ of size $O(nm)$, which makes the total ciphertext having the size of $O(nm)$.

Table 1: Comparison of several searchable encryption schemes. n is size of the document collection D, m is the size of the keyword space K, δ is the number of documents containing keywords of interest $\bar{K}$.

Scheme	Matching	Query Randomization	Security	Search Time	Index Size
Song *et al.* [1]	Exact	no	CPA	O(n)	N/A
Goh *et al.* [2]	Exact	no	CKA1	O(n)	O(n)
Curtmola *et al.*[9] (SSE-1)	Exact	no	CKA1	O(δ)	O(n+m)
Curtmola *et al.*[9] (SSE-2)	Exact	no	CKA2	O(δ)	O(nm)
Cao *et al.* [5]	Similarity	yes	CKA1	O(n)	O(nm)
Moataz *et al.* [15]	Exact	yes	CKA2	O(n)	O(nm)
Orencik *et al.* [16]	Exact	no	CKA2	O(n)	O(nm)
MKSim	**Similarity**	**yes**	**CKA2**	**O(δ)**	**O(nm)**

6.2 Performance

We have developed and implemented a multi-threaded proof-of-concept prototype of $MKSim$ scheme using C++ language. Our implementation includes 37 classes with a total of 10200 lines of code. Our prototype leverages two cryptographic libraries: *libtomcrypt* [3] and *HElib* [4]. *Libtomcrypt* is portable C cryptographic library that supports symmetric ciphers, one-way hashes, pseudo-random number generators, and a plethora of support routines. We use *libtomcrypt* to build the lookup filter T and encrypt the document collection. *HElib* is a C++ cryptographic library for homomorphic encryption available as an implementation of Brakerski *et al.* [36] scheme. We utilize *HElib* homomorphic cryptosystem since its one the few homomorphic toolkits that efficiently supports multiplication operation on the ciphertexts. Specifically, it provides the efficient "batching mode" property where a single ciphertext represents a vector of encrypted values. Thus, single homomorphic operation (i.e., multiplication) on two such ciphertexts applies the homomorphic operation component-wise to the entire vector. We use *HElib* to build the TF-IDF table Φ and compute the similarity measure of a search query and set of stored encrypted documents.

We show a thorough experimental evaluation of the MKSim scheme on a real-world dataset: the Internet Request for Comments (RFC) database [39], which is a collection of plaintext documents that consists of a large number of technical keywords describing different specifications, protocols, procedures and events in the Internet. All experiments have been performed on a 6 core Intel(R) Xeon(R) E5645 @ 2.40GHz processor and 98 GB memory running 64-bit Fedora 21 distro with the 3.19 kernel. We setup our prototype to use 20 pthread[40] threads in all experiments. The cloud server, data owner and data user applications were run on the same machine, as the network communication overhead was assumed to be negligible. In our evaluation we measure the computation and communication overheads of proposed solution. Specifically, we measure the overheads of the $BuildIndex$, $MakeQuery$ and $Evaluate$ algorithms presented in n Section 4.1.

Encryption Overhead. In this section, the overheads associated with the encryption process will be detailed. First, we study the size of keyword dictionary $K = (k_1, k_2, \ldots, k_m)$ extracted from RFC documents $D = (D_1, D_2, \ldots, D_n)$ of different sizes. Figure 2 shows the number of distinct keywords extracted from 250 to 2000 RFC documents. Figure 3 shows

[3]https://github.com/libtom/libtomcrypt

[4]https://github.com/shaih/HElib

that the storage overhead of keyword dictionary is nearly linear with the size of the dataset.

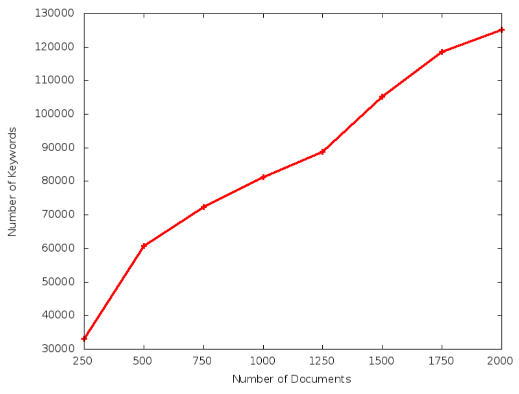

Figure 2: RFC Keyword Dictionary.

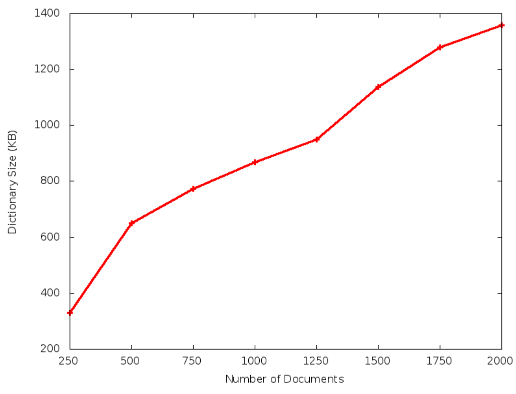

Figure 3: RFC Keyword Dictionary Size.

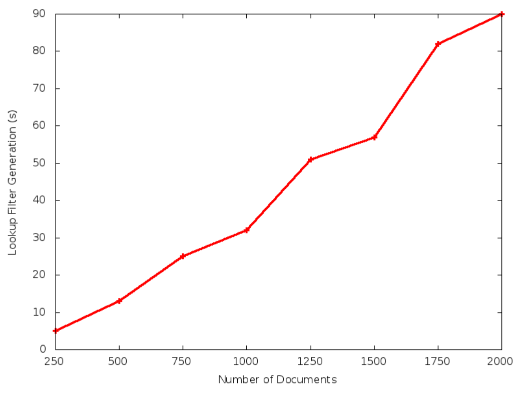

Figure 4: Lookup Filter T Generation.

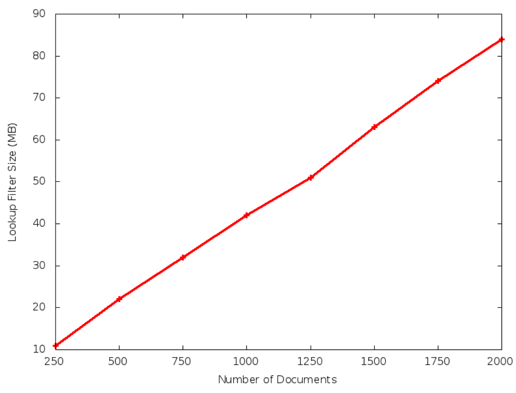

Figure 5: Lookup Filter T Size.

Second, we study computation and communication overheads for $BuildIndex$ algorithm. Our building index algorithm consists of two phases: constructing lookup filter T and TF-IDF table Φ. Figure 4 shows the time cost of constructing the lookup filter T using RFC documents of different sizes. The lookup filter T consists of applying a pseudo-random permutation on each keyword from the dictionary and labeling the output with document identifier where the keyword appeared. Figure 5 shows the storage requirements of lookup filter T.

Next, we measure the overhead related to TF-IDF table Φ. Our TF-IDF table Φ construction consists of computing the TFIDF table for each extracted keyword from RFC dataset and applying Brakerski's homomorphic cryptosystem in "batching mode" on each document row. We show the time cost of generating the TF-IDF table Φ in Figure 6 and the storage overhead in Figure 7. Although the time of building TF-IDF table Φ is not a negligible overhead for the data owner, this is a one-time operation before sending the encrypted documents to the cloud server.

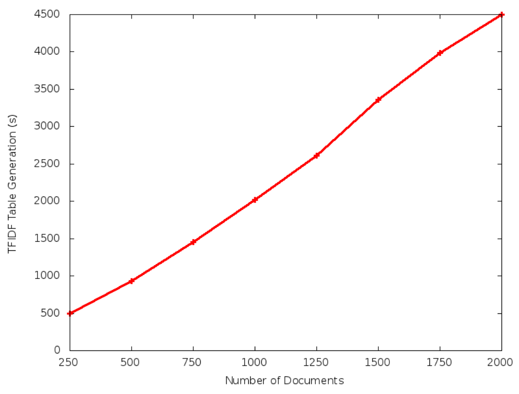

Figure 6: TF-IDF Table Φ Generation.

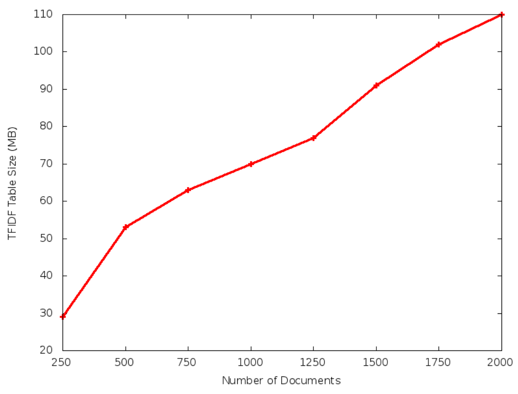

Figure 7: TF-IDF Table Φ Size.

Table 2: Search Queries.

Search Query 1 (SQ-1)	OSPF BGP MIME XML iSCSI
Search Query 2 (SQ-2)	Encryption Framework Certificate Authorization Authentication

Search Overhead. To evaluate the search overhead, we leverage two types of search inputs in our prototype. The first search query (SQ-1) describes five popular acronyms in the RFC dataset, while the second search query (SQ-2) is a set of general keywords describing technical security standards for the Internet. Table 2 shows the keywords used in our experiments.

First, we measure the performance of the $MakeQuery$ algorithm. This overhead of $MakeQuery$ consists of calculating a pseudo-random permutation applied on each queried keyword and calculating Brakerski's homomorphic encryption on each weighted keyword. Figure 8 shows the time cost for generating trapdoors for both SQ-1 and SQ-2 using keyword dictionaries of different sizes. Our results demonstrate that time cost to construct both SQ-1 and SQ-2 search queries stays mostly constant.

Second, we test the overhead of the $Evaluate$ algorithm. The $Evaluate$ execution at the cloud server consists of calculating similarity scores (i.e., homomorphic dot product) between a given search query and stored searchable index I. Table 3 shows the experimental results for RFC documents of different sizes. For each search input we measure the number of documents that match the lookup filter T and the time cost of calculating similarity scores for a set of selected documents. For example, for SQ-1 search input and RFC collection of 1500 documents, there are 472 documents that match the lookup filter T and it takes a total of 1427 seconds to measure similarity scores of selected 472 documents. Our results demonstrate that lookup filter T allows the cloud server to efficiently select the set of documents that contain the queried set of keywords without the need of calculating similarity scores for all documents in the dataset.

Third, using the results from Table 3 we measure the time cost to calculate the similarity score for each selected document. We define the Similarity Search Performance (SSP) metric as follows: $SSP(Q) = \frac{t_\Phi}{n_{\bar{K}}}$, where t_Φ corresponds to total time cost to calculate similarity scores for selected documents in Φ and $n_{\bar{K}}$ corresponds to the number of documents that match the search query in T. Figure 9 shows the SSP metric for both SQ-1 and SQ-2 search

Table 3: Search Query 1 and Search Query 2 Execution Results.

	Search Query 1		Search Query 2	
Number of Documents	Number of Matching Documents in T	Execution Time in Φ (s)	Number of Matching Documents in T	Execution Time in Φ (s)
250	85	217	152	341
500	173	454	307	738
750	255	680	462	1169
1000	324	873	608	1618
1250	400	1107	753	2068
1500	472	1427	919	2738
1750	562	1807	1071	3429
2000	626	2089	1220	3989

Table 4: Different Size Query Execution Results.

	1000 Documents		2000 Documents	
Search Query Size	Number of Matching Documents in T	Execution Time in Φ (s)	Number of Matching Documents in T	Execution Time in Φ (s)
5	11	57	123	518
10	222	643	1069	3634
25	286	807	1080	3694
50	340	924	1131	3867
100	571	1537	1172	3998
250	916	2424	1418	4880
500	959	2543	1892	6530
1000	988	2605	1950	6609

queries. Our results demonstrate that the cloud server has an average overhead of 2.8 seconds on each similarity score calculation. Also, Table 4 shows the experimental results for search queries of different sizes. We use two collections of sizes 1000 and 2000 documents and varied the size of search query. The results show that our search is proportional to the number of documents that contain a set of keywords of interest. In short, we notice that the MKSim protocol, although uses homomorphic encryption, adds minimal overhead on computation and storage. We believe that proposed solution can be easily deployed in a real-world cloud environment.

7 Group Multi-keyword Similarity Searchable Encryption (GMKSim)

In order to make an important step toward widespread use of searchable encryption, existing solutions need to include mechanisms to efficiently support hundreds even thousands cloud users in the system[41]. In this section we consider a multi-user scenario that involves a data owner that wishes to share a documents and multiple data users that want to query encrypted data using the cloud server.

Curtmola *et al.* [9] were the first to extend their single-user scheme with broadcast encryption[42],

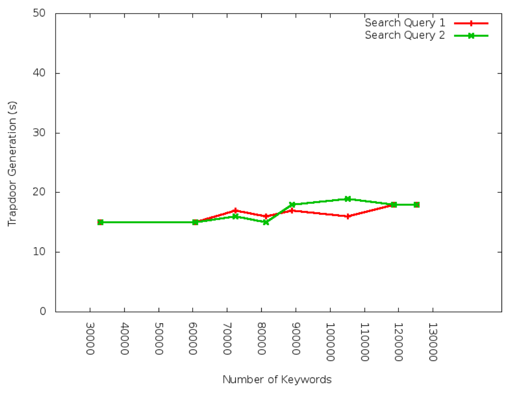

Figure 8: Time Cost of Generating Trapdoor.

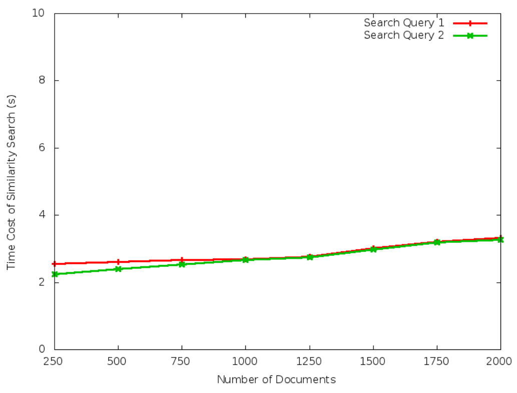

Figure 9: Cloud Similarity Search Time.

where the data owner is able to outsource an encrypted document collection to the cloud server and an arbitrary group of users is allowed to query the data. Broadcast encryption allows the data owner to distribute a shared secret key to a group of data users. However, this solution might not work in a real-world cloud deployment that involves potentially large number of on-demand data users since only one key is shared among all users and each user revocation requires a new key to be distributed to the remaining users. It is desirable that each data user could keep its own secret key, which makes key management easier and more efficient.

We propose a new Group Multi-keyword Similarity Searchable Encryption (GMKSim) scheme that solves the problem of managing access privileges and searching multiple keywords over encrypted cloud data. Our extension is based on distributed broadcast encryption scheme[44] that supports a distributed setup where the data owner sets up the system, but each user generates their own key when joining the system. In fact, we remove the burden of key management from the data owner and let group establishment run by participating data users (i.e., data users are allowed to pick desired participants and establish a shared key to search remote encrypted data). We first give the set of definitions and security notions of GMKSim scheme. Later we present an efficient construction of GMKSim that combines the idea of a single-user MKSim scheme with distributed broadcast encryption scheme.

7.1 Preliminaries

We begin with the definition of *witness pseudo-random function* (WPRF). Informally, a witness PRF for an NP language L is a PRF F such that anyone with a valid witness that $x \in L$ can compute $F(x)$ without the secret key, but for all $x \notin L$, $F(x)$ is computationally hidden without knowledge of the secret key. Formally, a witness PRF is defined as follows:

Definition 15. Witness Pseudo-Random Function (WPRF)[44]: A witness PRF is a triple of algorithms $(Gen, F, Eval)$ such that

- Gen: a probabilistic algorithm that inputs a security parameter λ and a circuit $R : \mathbb{X} \times \mathbb{W} \to \{0, 1\}$, and outputs a secret function key fk and a public evaluation key ek.
- F: a deterministic algorithm that inputs the function key fk and an input $x \in \mathbb{X}$, and outputs some output $y \in \mathbb{Y}$ for some set $\mathbb{Y}$.

- $Eval$: a deterministic algorithm that inputs the evaluation key ek, an input $x \in \mathbb{X}$ and a witness $w \in \mathbb{W}$, and produces an output $y \in \mathbb{Y}$ or $\perp$.

The following correctness property is required to hold:

$$Eval(ek, x, w) = \begin{cases} F(fk, x) & \text{if } R(x, w) = 1 \\ \perp & \text{if } R(x, w) = 0 \end{cases} \text{ for all } x \in \mathbb{X}, w \in \mathbb{W}.$$

A multiparty key exchange protocol allows a group of g users to simultaneously post a message to a public bulletin board, retaining some user-independent secret. After reading off the contents of the bulletin board, all users establish the same shared secret key. The multiparty key exchange protocol consists of the following algorithms:

Definition 16. Non-Interactive Multiparty Key Exchange protocol (NIKE-WPRF)[44]: Let $G : \mathbb{S} \rightarrow \mathbb{Z}$ be a pseudo-random generator with $|S|/|Z| \leq negl$. Let $WPRF = (Gen, F, Eval)$ be a witness PRF. Let $R_g : \mathbb{Z}^g \times (\mathbb{S} \times [g]) \rightarrow \{0, 1\}$ be a relation that outputs 1 on input $((z_1, \ldots, z_g), (s, i))$ if and only if $z_i = G(s)$. The non-interactive key exchange protocol consists of:

- $Publish(\lambda, g)$: a probabilistic algorithm that takes as input the security parameter λ and the group order g. It computes $(fk, ek) \overset{R}{\leftarrow} Gen(\lambda, R_g)$. Next, it picks a random seed $sk \overset{R}{\leftarrow} \mathbb{S}$ and compute $z \leftarrow G(sk)$. It outputs a secret key sk and public values (z, ek), where sk is kept secret and (z, ek) are published to the bulletin board.
- $KeyGen(\{z_i, ek_i\}_{i \in [g]}, sk)$: a deterministic algorithm that inputs group g and user's secret sk. It outputs a group key $k = Eval(ek_i, (z_1, \ldots, z_g), (sk, i))$.

Broadcast encryption[42] allows an encryptor (data owner) broadcast a message to a subset of recipients (data users). The system is said to be collusion resistant if non-data users can learn information about the plaintext. Boneh *et al.*[43] recently proposed a new distributed broadcast encryption that is based on NIKE where data users generate secret keys on their own and simply append their corresponding public values to the broadcast public key. Unlike existing broadcast encryption schemes[45, 46, 47] where participants are assigned their secret key by a trusted authority (data owner), distributed broadcast scheme has no trusted authority and each user generates a secret key for itself. However, scheme in [43] is based on the indistinguishable obfuscation, for which none practical construction is known[44]. Most recent work by Zhandry *et al.* [44] proposes a distributed broadcast encryption scheme that replaces indistinguishable obfuscators with a witness PRFs. The scheme is simpler and much more efficient that current obfuscation candidates. We use the following definition of distributed broadcast encryption scheme:

Definition 17. Distributed Broadcast Encryption over NIKE (BE-NIKE-WPRF) [44]: Distributed broadcast encryption scheme over multi-party non-interactive key exchange protocol consists of four following algorithms:

1. $Setup$: a probabilistic algorithm to setup BE-NIKE-WPRF scheme. The algorithm outputs a secret parameter λ and group order g.
2. $Join(\lambda, g)$: a probabilistic algorithm to join the scheme that is executed by each participant. The algorithm inputs a secret parameter λ and group order g. The algorithm invokes *NIKE-WPRF*.$Publish(\lambda, g)$ to output secret sk and public values (z, ek). The user makes (z, ek) publicly available to other participants.

3. $Enc(\{z_i, ek_i\}_{i\in[g]}, sk, m)$: a probabilistic algorithm to encrypt message m under shared key. The algorithm inputs the set of public values $\{z_i, ek_i\}_{i\in[g]}$, secret key sk and plaintext message m. The algorithm runs *NIKE-WPRF.*$KeyGen(\{z_i, ek_i\}_{i\in[g]}, sk)$ to derive the shared key k. The algorithm outputs a ciphertext c which is the encryption of message m using the shared key k.

4. $Dec(\{z_i, ek_i\}_{i\in[g]}, sk, c_m)$: a deterministic algorithm to decrypt c_m. The algorithm invokes *NIKE-WPRF.*$KeyGen(\{z_i, ek_i\}_{i\in[g]}, sk)$ to derive k. If $k \neq \perp$, then algorithm decrypts c_m using k and outputs the original message m.

This completes cryptographic preliminaries used in our solution. We are now ready to present the GMKSim scheme.

7.2 Algorithm Definitions

Definition 18. Group Multi-keyword Similarity Searchable Encryption (GMKSim): An index-based GMKSim scheme over a set of documents D is a tuple of eight polynomial-time algorithms $GMKSim = (Gen, BuildIndex, Join, SetupGroup, Revoke, MakeQuery, Evaluate, Decrypt)$, as follows:

1. $(S_1, S_2, PK, SK, \lambda, g) \leftarrow Gen(1^s)$: a probabilistic algorithm run by the data owner to setup the GMKSim scheme. The algorithm invokes $MKSim.Gen$ with an input of a secret parameter s, and outputs a set of keys S_1, S_2, PK, SK. The algorithm runs *BE-NIKE-WPRF.*$Setup$ to output secret parameter λ and group order g.

2. $(I, C) \leftarrow BuildIndex(S_1, S_2, PK, D, K)$: a probabilistic algorithm run by the data owner to encrypt a document collection D. The algorithm invokes $MKSim.BuildIndex$ with an input of keys S_1, S_2 and PK, a document collection D and a keyword dictionary K. It outputs a searchable index I and a set of encrypted documents C.

3. $(sk, (z, ek)) \leftarrow Join(\lambda, g)$: a probabilistic algorithm run by each data user to participate in the scheme. The algorithm invokes *BE-NIKE-WPRF.*$Join$ with an input of secret parameter λ and group order g. It outputs a pair $(sk, (z, ek))$.

4. $c_r \leftarrow SetupGroup(\{z_i, ek_i\}_{i\in[h]}, sk)$: a probabilistic algorithm run by the group owner to establish the group $h \subseteq g$ of authorized data users. The algorithm runs *BE-NIKE-WPRF.*Enc with an input of public values $\{z_i, ek_i\}_{i\in[h]}$, group owner's secret key sk and a sampled secret r. The output is encrypted ciphertext c_r.

5. $c_r \leftarrow Revoke(\{z_i, ek_i\}_{i\in[h\setminus o],}, sk)$: a probabilistic algorithm run by the group owner to remove a user o from the set of authorized users. The algorithm invokes *BE-NIKE-WPRF.*Enc that inputs the set of public values $\{z_i, ek_i\}_{i\in[h\setminus o]}$, group owner's secret key sk and a new secret r. The output is encrypted ciphertext c_r.

6. $\Omega \leftarrow MakeQuery(S_2, PK, K, \bar{K}, c_r)$: a probabilistic algorithm run by a data user to construct a search query. The algorithm invokes *BE-NIKE-WPRF.*Dec with an input of public values $\{z_i, ek_i\}_{i\in[h]}$, secret key sk and ciphertext c_r. It outputs a secret r. If $r \neq \perp$, the algorithm invokes $MKSim.MakeQuery$ with keys S_2, PK, keyword dictionary K, and set of keywords of interest $\bar{K}$. The output is a search query Ω encrypted under secret r.

7. $L \leftarrow \mathit{Evaluate}(PK, I, \Omega, C, c_r)$: a deterministic algorithm run by the cloud server. The algorithm invokes *BE-NIKE-WPRF.Dec* an input of public values $\{z_i, ek_i\}_{i\in[h]}$, secret key sk and ciphertext c_r, and it outputs secret r to decrypt search query Ω. The algorithm runs $\mathit{MKSim.Evaluate}$ with an input of public key PK, searchable index I, search query Ω and set of encrypted documents C. The algorithm outputs a sequence of identifiers $L \subseteq C$.

8. $D_i \leftarrow \mathit{Decrypt}(S_1, SK, L_i)$: a deterministic algorithm that inputs a set of secret keys S_1, SK and an encrypted document L_i. The algorithm outputs a document D_i.

We now formalize the security of proposed scheme.

- We require that the cloud server should not learn anything about the documents: an adversary with an access to searchable index I and encrypted document collection $C = (C_1, C_2, \dots, C_n)$ should learn nothing about original documents $D = (D_1, D_2, \dots, D_n)$
- We require that the cloud server should not learn anything from the search queries beyond the access and search patterns: an adversary with an access to a search queries $(Q_1, Q_2, \dots, Q_m)$ generated by the data user learns nothing about the content of each search query Q_i or the content of resulted documents.
- We require the revocation for the data users: once a data user removed from the set of authorized data users, he/she is no longer able to execute a search over encrypted documents.

In GMKSim we use the adaptive semantic security notion of a single-user MKSim scheme. It provides the security against an adaptive adversary: cloud server does not learn anything about the documents and search queries beyond the access and search patterns. However, with an addition of access privilege property, we need to expand our security definitions towards the *Revoke* algorithm. We define the probabilistic experiment *Rev* as follows:

Definition 19. Revocation: Let $GMKSim = (\mathit{Gen}, \mathit{BuildIndex}, \mathit{Join}, \mathit{SetupGroup}, \mathit{Revoke}, \mathit{MakeQuery}, \mathit{Evaluate}, \mathit{Decrypt})$ be a group MKSim scheme, s be a security parameter, and $A = (A_1, A_2, A_3)$ be an adversary. We use the following probabilistic experiment $Rev_{GMKSim,A}$(s):

```
$Rev_{GMKSim,A}(s)$:
$S_1, S_2, PK, SK, \lambda, g \leftarrow Gen(1^s)$
$(st_A, D, K) \leftarrow A_1(1^s)$
$(sk_A, (z_A, ek_A)) \leftarrow Join(\lambda, g)$
$c_r \leftarrow SetupGroup((z_A, ek_A), sk)$
$(I, C) \leftarrow BuildIndex(S_1, S_2, PK, D, K)$
$st_A \leftarrow A_2^{O(I,C,st_{\mathbb{S}},\cdot)}(st_A, sk_A, (z_A, ek_A), c_r)$
$c_r^{'} \leftarrow Revoke((z_A, ek_A), sk)$
$\Omega \leftarrow A_3(st_A)$
$L \leftarrow Evaluate(st_S, PK, I, \Omega, C, c_r^{'})$
if $L \neq \perp$, output 1, otherwise output 0,
```

where $O(I, C, st_{\mathbb{S}}, \cdot)$ is an oracle that inputs a search query Ω and outputs ciphertexts C indexed by $L \leftarrow \mathit{Evaluate}(PK, I, \Omega, C, c_r^{'})$ if $L \neq \perp$ and $\perp$ otherwise. We claim that *Revoke*

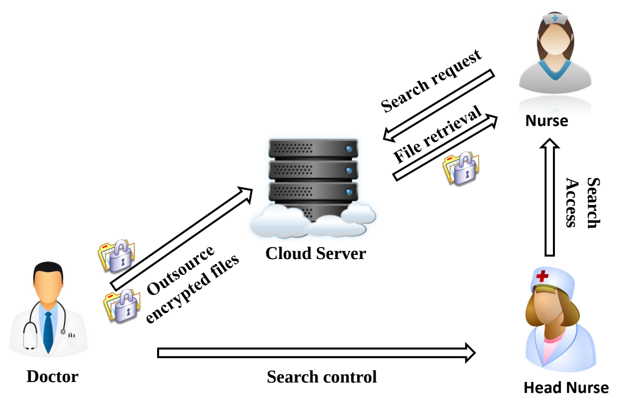

Figure 10: Group Hospital Example.

algorithm achieves user revocation if for all polynomial-size adversaries $A = (A_1, A_2, A_3)$ the following is correct:

$$Pr[Rev_{GMKSim,A}(s) = 1] \leq negl(s), \tag{10}$$

where the probability is over the coins of Gen, $Join$, $SetupGroup$, $Revoke$ and $BuildIndex$.

7.3 GMKSim Construction

Figure 11 shows the details of $GMKSim = (Gen, BuildIndex, Join, SetupGroup, Revoke, MakeQuery, Evaluate, Decrypt)$. In our construction we combine the ideas of a single-user $MKSim = (Gen, BuildIndex, MakeQuery, Evaluate, Decrypt)$ and a distributed broadcast encryption scheme $BE\text{-}NIKE\text{-}WPRF = (Setup, Join, Enc, Dec)$. We require standard security notions for broadcast encryption. Specifically, in addition of providing PCPA-security, it provides revocation-scheme security against a group of all revoked users.

Our construction relies on a pseudo-random permutation $\rho : \{0,1\}^r \times \{0,1\}^t \to \{0,1\}^t$, where r is the secret parameter and t is the size of search query Ω in the $MKSim$ scheme. We assume the honest-but-curious adversarial model for the cloud server. We also assume that the cloud server does not collude with revoked users, otherwise our construction cannot prevent a revoked user from executing a search.

To describe $GMKSim$ in details we use following hospital example illustrated in Figure 10. Consider a doctor (data owner) that performed a blood test on a patient and wishes to share resulted documents with group of nurses (data users) in the hospital. The doctor considers building a searchable index I from the resulted documents D, and sending both index I and encrypted results C to the hospital blackboard running on the cloud server. To remove the burden of key management, the doctor enables distributed setup, where each nurse generates its own secret key (when joining the system) and establishes a group of authorized participants (e.g., a head nurse includes her subordinate nurses). The doctor begins with sampling a secret parameter s, which is used as input to generate the set of

```
$\underline{Gen(1^s):}$
    1. generate $S_1$, $S_2$, $PK$, $SK \leftarrow MKSim.Gen(1^s)$.
    2. generate $\lambda$, $g \leftarrow BE\text{-}NIKE\text{-}WPRF.Setup(1^s)$.
Output the key set $S_1$, $S_2$, $PK$, $SK$, secret parameter $\lambda$ and group order $g$.
$\underline{BuildIndex(S_1, S_2, PK, D, K):}$
    1. set $(I, C) \leftarrow MKSim.BuildIndex(S_1, S_2, PK, D, K)$.
Store $(I, C)$ to the cloud server.
$\underline{Join(\lambda, g):}$
    1. generate $(sk, (z, ek)) \leftarrow BE\text{-}NIKE\text{-}WPRF.Join(\lambda, g)$.
Keep $sk$ private, output $(z, ek)$ to the cloud server.
$\underline{SetupGroup(\{z_i, ek_i\}_{i\in[h]}, sk}$
    1. pick $h \subseteq g$ and get public values $\{z_i, ek_i\}_{i\in[h]}$ from the cloud server.
    2. sample $r \leftarrow \{0,1\}^s$ and compute $c_r \leftarrow BE\text{-}NIKE\text{-}WPRF.Enc(\{z_i, ek_i\}_{i\in[h]}, sk, r)$.
Output $c_r$ to the cloud server.
$\underline{Revoke(\{z_i, ek_i\}_{i\in[h\backslash o]}, sk)}$
    1. set $(h\backslash o) \subseteq g$ and retrieve public values $\{z_i, ek_i\}_{i\in[h\backslash o]}$ from the cloud server.
    2. sample new $r \leftarrow \{0,1\}^s$ and compute $c_r \leftarrow BE\text{-}NIKE\text{-}WPRF.Enc(\{z_i, ek_i\}_{i\in[h\backslash o]}, sk, r)$.
Output new $c_r$ to the cloud server.
$\underline{MakeQuery(S_2, PK, K, \bar{K}, c_r):}$
    1. retrieve $c_r$ and from the cloud server.
    2. compute $r \leftarrow BE\text{-}NIKE\text{-}WPRF.Dec(\{z_i, ek_i\}_{i\in[h]}, sk, c_r)$. If $r = \perp$, output $\perp$.
    3. calculate $\Omega' \leftarrow MKSim.MakeQuery(S_2, PK, K, \bar{K})$.
    4. set $\Omega \leftarrow \rho(r, \Omega')$.
Output $\Omega$.
$\underline{Evaluate(PK, I, \Omega, C, c_r):}$
    1. compute $r \leftarrow BE\text{-}NIKE\text{-}WPRF.Dec(\{z_i, ek_i\}_{i\in[h]}, sk, c_r)$.
    2. compute $\Omega' \leftarrow \rho^{-1}(r, \Omega)$.
    3. output $L \leftarrow MKSim.Evaluate(PK, I, \Omega', C)$, where $L \in C$.
Output $L$.
$\underline{Decrypt(S_1, SK, L_i):}$
Output $D_i \leftarrow MKSim.Decrypt(S_1, SK, L_i)$.
```

Figure 11: GMKSim Construction.

secret keys. The key generation algorithm outputs a set of keys S_1, S_2, PK, SK for a single-user scheme, secret key λ and group order g for a distributed broadcast encryption scheme. The doctor next invokes the $BuildIndex$ algorithm of a single-user scheme $MKSim$ outlined in Figure 1. The outputs are searchable index I and set of encrypted documents C. Note, the searchable index I consists of a lookup filter T, based on SSE-2 construction, and a TF-IDF table Φ constructed using a term weight importance. Next, each nurse launches the $Join$ algorithm with secret λ and group g (both distributed by the doctor) to generate an output of $(sk, (z, ek)$. Secret key sk is kept private, while (z, ek) are published to the cloud server.

Now, the head nurse (group owner) creates a group of authorized users (other participants) that are allowed to execute search over encrypted cloud data. The head nurse launches the $SetupGroup$ algorithm where she picks public values $\{z_i, ek_i\}_{i\in h}$ of authorized participants $h \subseteq g$, samples random secret parameter r, and uses the distributed broadcast encryption to output the ciphertext c_r. Finally, head nurse sends c_r to the cloud server. Now the document collection is available for selective retrieval.

To search for a keywords of interest $\bar{K}$, a nurse (data user) executes $MakeQuery$ algorithm that includes four steps. First, she contacts the cloud server and retrieves the ciphertext c_r. Next, she invokes the distributed broadcast encryption algorithm with her own secret sk, public values $\{z_i, ek_i\}_{i \in h}$ to recover the secret r. If r successfully recovered, she inputs keywords $\bar{K}$ to a single-user *MKSim*.$MakeQuery$ algorithm that outputs search query Ω'. Next, the nurse forms a permuted search query by applying PRP ρ with secret key r, i.e. $\Omega = \rho(r, \Omega)$ which is sent to the cloud server. The cloud server, upon receiving Ω, recovers the search query by executing $\rho^{-1}(r, \Omega)$. Here, the key r used in ρ is known only by the head nurse, the cloud server and the set of authorized data users h. Once Ω' is decrypted, the cloud server executes a single-user *MKSim*.$Evaluate$ algorithm to find a set of encrypted documents that match $\bar{K}$ keywords of interest.

If a nurse o is no longer an authorized user in group, the head nurse invokes the $Revoke$ algorithm on o. Specifically, the head nurse samples a new secret r' and generate a new cloud server ciphertext $c_r^{'}$ using distributed broadcast encryption algorithm that exclude the public values (z, ek) of nurse o. The new cloud ciphertext $c_r^{'}$ is distributed to the cloud server to replace the old c_r. Since revoked data user o is not able to recover the new secret r' in the $MakeQuery$ algorithm, permuted Ω will not yield a valid search query after applying $\rho^{-1}(r^{'}, \Omega)$ permutation at the cloud server. This simple extra layer given by the pseudo-random permutation ρ prevents data users from performing successful search once they are removed from the system. Proposed solution is very efficient since the cloud server only needs to evaluate a pseudo-random permutation to decide whether a data user is authorized to search an encrypted document collection.

The following theorem shows that the $GMKSim$ scheme satisfies revocability.

Theorem 20. *The* $GMKSim = (Gen, BuildIndex, Join, SetupGroup, Revoke, MakeQuery, Evaluate, Decrypt)$ *achieves revocability according to Definition 19.*

Proof. The proof is straightforward, and we only state the intuition behind the proof. The revocation of proposed scheme relies on the following assumption. The PCPA-security of *BE-NIKE-WPRF* scheme guarantees that the simulated search query $\Omega^\star$ is indistinguishable from the real search query Ω. Indeed, this is true since decryption of the new message $c_r^{'}$ (generated by the group owner that excludes the public values of revoked user) will never output a valid r' that is used to generate the real Ω. $\square$

8 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 homomorphic dot product. 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. We also presented a group scheme which extends the original solution to support multiple parties. As future work, we plan to optimize the index construction algorithm of proposed solution. We also consider a system that allows ranking search as well as efficient secure updates on the encrypted cloud data. Furthermore, we consider a solution that is secure against malicious cloud servers and it allows data users to verify the correctness of returned results.

Acknowledgments

This work was partially supported by the U.S. National Science Foundation under Grant No. 0905232. We are grateful to two anonymous referees for their constructive comments and valuable suggestions that helped to improve this paper.

References

[1] Dawn Xiaodong Song, David Wagner, and Adrian Perrig: Practical techniques for searches on encrypted data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, 2000.

[2] Eu-Jin Goh: Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003.

[3] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano: Public Key Encryption with Keyword Search. In Proceedings of the 23rd Annual Eurocrypt Conference, 2004.

[4] Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 8, pp. 1467-1479, 2012.

[5] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In Proceedings of the 30th IEEE International Conference on Computer Communications, 2011.

[6] Marina Blanton, Mikhail J. Atallah, Keith B. Frikken, and Qutaibah M. Malluhi: Secure and efficient outsourcing of sequence comparisons. In Proceedings of the 17th European Symposium on Research in Computer Security, 2012.

[7] Jonathan Katz and Yehuda Lindell. Introduction to Modern Cryptography: Principles and Protocols. Chapman & Hall/CRC Cryptography and Network Security Series, 2007.

[8] Marina Blanton: Achieving full security in privacy-preserving data mining. In Proceedings of the 3rd IEEE International Conference on Privacy, Security, Risk and Trust, 2011.

[9] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky: Searchable symmetric encryption: improved definitions and efficient constructions. In Proceedings of the 13th ACM Conference on Computer and Communications Security, 2006.

[10] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill: Deterministic and efficiently searchable encryption. In Proceedings of the 27th Annual International Cryptology Conference, 2007.

[11] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malon e-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. Journal of Cryptology, vol. 21, no. 3, pp. 350-391, 2008.

[12] Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati: Private data indexes for selective access to outsourced data. In Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, 2011.

[13] Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu: Efficient similarity search over encrypted data. In Proceedings of the 28th IEEE International Conference on Data Engineering, 2012.

[14] Mikhail Strizhov and Indrajit Ray: Multi-keyword similarity search over encrypted cloud data. In Proceedings of the 29th International Conference on ICT Systems Security and Privacy Protection, 2014.

[15] Tarik Moataz and Abdullatif Shikfa: Boolean symmetric searchable encryption. In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, 2013.

[16] Cengiz Orencik, Murat Kantarcioglu, and Erkay Savas: A practical and secure multi-keyword search method over encrypted cloud data. In Proceedings of the 6th IEE International Conference on Cloud Computing, 2013.

[17] Seny Kamara, Charalampos Papamanthou, and Tom Roeder: Dynamic searchable symmetric encryption. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, 2012.

[18] Cong Wang, Kui Ren, Shucheng Yu, and Karthik Mahendra Raje Urs: Achieving usable and privacy-assured similarity search over outsourced cloud data. In Proceedings of the 31st Annual IEEE International Conference on Computer Communications, 2012.

[19] Alexandra Boldyreva and Nathan Chenette: Efficient fuzzy search on encrypted data. In Proceedings of the 21st International Workshop on Fast Software Encryption, 2014.

[20] Oded Goldreich: The foundations of cryptography: volume 2, basic applications. Cambridge University Press, 2004.

[21] Giovanni Di Crescenzo and Vishal Saraswat: Public key encryption with searchable keywords based on jacobi symbols. In Proceedings of the 8th International Conference on Cryptology in India, 2007.

[22] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner: Highly-scalable searchable symmetric encryption with support for boolean queries. In Proceedings of the 33rd Annual International Cryptology Conference CRYPTO 2013, 2013.

[23] Oded Goldreich and Rafail Ostrovsky: Software protection and simulation on oblivious RAMs. Journal of the ACM, vol. 43, no. 3, pp. 431-473, 1996.

[24] Mikhail J. Atallah, Florian Kerschbaum, and Wenliang Du: Secure and Private Sequence Comparisons. In Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society, 2003.

[25] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, Srinivas Devadas: Path ORAM: an extremely simple oblivious RAM protocol. In Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, 2013.

[26] Tarik Moataz, Travis Mayberry, Erik-Oliver Blass, and AgesHui Chan: Resizable tree-based oblivious RAM. In Proceedings of the 19th International Conference on Financial Cryptography and Data Security, 2015.

[27] Muhammad Naveed: The Fallacy of Composition of Oblivious RAM and Searchable Encryption. Cryptology ePrint Archive, Report 2015/668, 2015.

[28] Mikhail Strizhov and Indrajit Ray: Substring position search over encrypted cloud data using tree-based index. In Proceedings of the 2015 IEEE International Conference on Cloud Engineering, 2015.

[29] Craig Gentry: A fully homomorphic encryption scheme. Stanford University, 2009.

[30] Craig Gentry: Fully Homomorphic Encryption Using Ideal Lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, 2009.

[31] David A. Grossman and Ophir Frieder: Information Retrieval: Algorithms and Heuristics. Springer, 2004.

[32] Justin Zobel and Alistair Moffat: Exploring the Similarity Space. SIGIR Forum, vol. 32, no. 1, pp. 18-34, 1998.

[33] Ian H. Witten, Alistair Moffat, and Timothy C. Bell: Managing Gigabytes (2nd Ed.): Compressing and Indexing Documents and Images. Morgan Kaufmann Publishers Inc., 1999.

[34] Zvika Brakerski: Fully homomorphic encryption without modulus switching from classical GapSVP. In Proceeding of the 32nd Annual International Cryptology Conference CRYPTO 2012, 2012.

[35] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan: (Leveled) Fully homomorphic encryption without bootstrapping. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, 2012.

[36] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan: Fully Homomorphic Encryption without Bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011.

[37] Yan-Cheng Chang and Michael Mitzenmacher: Privacy preserving keyword searches on remote encrypted data. In Proceedings of the 3rd International Conference on Applied Cryptography and Network Security, 2005.

[38] Ashwin Swaminathan, Yinian Mao, Guan-Ming Su, Hongmei Gou, Avinash L. Varna, Shan He, Min Wu, and Douglas W. Oard: Confidentiality-preserving rank-ordered search. In Proceedings of the ACM Workshop on Storage Security and Survivability, 2007.

[39] RFC: Request for comments database. http://www.ietf.org/rfc.html, 2015.

[40] Bradford Nichols, Dick Buttlar, and Jacqueline Proulx Farrell: Pthreads programming. O'Reilly & Associates, Inc., 1996.

[41] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter: A survey of provably secure searchable encryption. ACM Computing Surveys, vol. 47, no. 2, pp. 1-51, 2014.

[42] Amos Fiat and Moni Naor: Broadcast encryption. In Proceedings of the 13th Annual International Cryptology Conference CRYPTO 1993, 1993.

[43] Dan Boneh and Mark Zhandry: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Proceedings of the 34th Annual International Cryptology Conference CRYPTO 2014, 2014.

[44] Mark Zhandry: How to Avoid Obfuscation Using Witness PRFs. In Proceedings of the 13th IACR Theory of Cryptography Conference TCC 2016, 2016.

[45] Dan Boneh, Craig Gentry, and Brent Waters: Collusion resistant broadcast encryption with short ciphertexts and private keys. In Proceedings of the 25th Annual International Cryptology Conference CRYPTO 2005, 2005.

[46] Ryuichi Sakai and Jun Furukawa: Identity-based broadcast encryption. Cryptology ePrint Archive, Report 2007/217, 2007.

[47] Cecile Delerablee, Pascal Paillier, and David Pointcheval: Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In Proceedings of the First International Conference on Pairing-based Cryptography, 2007.