A Practical and Secure Multi-Keyword Search Method over Encrypted Cloud Data

Cengiz Orencik*, Murat Kantarcioglu† and Erkay Savas*
*Faculty of Engineering and Natural Sciences
Sabanci University, Istanbul, 34956, Turkey

†Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080, USA

***Abstract*—Cloud computing technologies become more and more popular every year, as many organizations tend to outsource their data utilizing robust and fast services of clouds while lowering the cost of hardware ownership. Although its benefits are welcomed, privacy is still a remaining concern that needs to be addressed. We propose an efficient privacy-preserving search method over encrypted cloud data that utilizes *minhash* functions. Most of the work in literature can only support a single feature search in queries which reduces the effectiveness. One of the main advantages of our proposed method is the capability of multi-keyword search in a single query. The proposed method is proved to satisfy adaptive semantic security definition. We also combine an effective ranking capability that is based on term frequency-inverse document frequency (tf-idf) values of keyword document pairs. Our analysis demonstrates that the proposed scheme is proved to be privacy-preserving, efficient and effective.**

I. Introduction

Due to increasing storage and communication requirements, today's organizations demonstrate a strong tendency to outsource their searchable data to remote servers. Clouds provide efficient and cost effective solutions for data storage and data processing requirements of organizations. Nevertheless, the outsourced data may contain sensitive information that needs to be hidden. An essential requirement, with which the cloud providers are not necessarily trusted. Therefore, some precautions are required to protect the sensitive data from both the cloud server and any other non-authorized party.

One of the most important operations on remote data is the search operation. The data is assumed to be accessible to several authorized users that frequently execute search operations. Hence, the search operation should not only protect the privacy of the users and the data but also should be highly efficient. Due to the significance of privacy concerns, privacy-preserving search methods have been extensively studied in recent years. While most of these work focus on single keyword search [1], [2], [3], [4], few propose solutions to multi-keyword search [5], [6], [7]. As we show in the experiments section (cf. Section VIII), our proposed work provides a significantly more efficient solution than [5], [6], [7]. Considering the large dataset sizes, a single keyword search query usually matches with lots of data items, where only few are relevant. Moreover, user needs to apply several queries and takes the intersection of the corresponding results, which imposes a serious burden of both computation and time on the user. A multi-keyword search, instead can incorporate a conjunction of several keywords in a single query. Via increasing the search constraints, only the most relevant items will be returned to the user which reduces the computation burden on the users. Therefore, in this work, we propose a novel secure and efficient multi-keyword search method that returns the matching data items in a ranked ordered manner.

The contributions of this paper are multifold. Firstly, we present a novel *minhash* based privacy-preserving multi-keyword search method that provides high precision rates. Secondly, we provide security requirements and formally prove that the proposed method satisfies adaptive semantical security. Thirdly, we utilize a ranking method based on term frequencies and inverse document frequencies (tf-idf) of keywords. Finally, we implement the proposed scheme and demonstrate that it is efficient and effective by providing the implementation results.

The rest of this work is organized as follows. In Section II, we discuss the related work. The preliminary background information such as *minhash* functions and tf-idf values is given in Section III. In Section IV, we provide the framework of the proposed model and define the necessary security terms and requirements. Then, we present the crucial steps of our proposed method in Section V. We formally prove that the privacy-preserving scheme we propose is adaptive semantically secure in Section VI. In Section VII, we propose an improvement to our scheme utilizing multiple servers. An extensive cost analysis and comparison of the proposed method with the most related work are given in Section VIII. Finally, Section IX is devoted for the concluding remarks.

II. Related Work

The problem of privacy-preserving keyword search is addressed by various work in literature. Related work can be analyzed in two major groups: single keyword and multi keyword search. While the user can only search for a single feature per query in the former, the latter enables search for a conjunction of several keywords in a single query.

Most of the privacy-preserving keyword search protocols existing in literature concentrate on single keyword search.

Goh [8] proposes a security definition for formalization of the security requirements of searchable symmetric encryption schemes. One of the first privacy-preserving search protocols is proposed by Ogata and Kurosawa [1] using RSA blind signatures. The scheme is not very practical due to the heavyweight public key operations per database entry that should be performed on the user side.

Later, Curtmola [3] provides adaptive security definitions for privacy-preserving keyword search protocols and proposes a scheme that satisfies the requirements given in the definitions. Another single keyword search scheme is proposed by Wang et al. [2] that keeps an encrypted inverted index together with relevancy scores for each keyword document pair. Different from the previous work, this method is capable of ranking the results according to their relevancy with the search term.

Recently, Kuzu et al. [4] propose another single keyword search method that uses locality sensitive hashes (LSH) and satisfies adaptive semantic security. Different from the other work, this scheme is a similarity search scheme, which means that matching algorithm works even some typos exist in the query. We take the locality sensitive hashing idea used in [4] for **single keyword search** and adapt it to efficient **multi-keyword search**.

All the work that are given above, are only capable of conducting single keyword search. However, in the typical case of search over encrypted cloud data, the size of the outsourced dataset is usually huge and single keyword search will inevitably return an excessive number of matches where most will be irrelevant for the user. Multi-keyword search allows more constraints in the search query and enables the user to access only the most relevant data. Raykova et al. [9] proposed a solution using a protocol called re-routable encryption. They introduce a new agent called query router (QR) between the user and the server. User sends the queries to the server through this QR to protect his anonymity with respect to the server. Security of the user's message with respect to QR is satisfied by confidentiality (i.e., encryption). They utilize bloom filters for efficient search. Although this work is presented as a single keyword search method, the authors also show a trivial multi-keyword extension. Wang et al. [10] proposed a multi keyword search scheme, which is secure under the random oracle model. The method uses a hash function to map keywords into a fixed length binary array. Later an improvement to this work is proposed by Orencik and Savas [6] that additionally provides strict privacy protection and ranking capability. Cao et al. [7] proposed another multi keyword search scheme that encodes the searchable database index into two binary matrices and uses inner product similarity during matching. This method requires keyword fields in the index. This means that the user must know a list of all valid keywords and their positions as a compulsory information to generate a query. This assumption may not be applicable in several cases. While our work is more efficient than [9], [7], the privacy requirements that we satisfy is stricter compared to the ad-hoc solutions in [10], [6]. Detailed comparative analysis is provided in Section VIII.

Bilinear pairing based solutions for privacy-preserving multi-keyword search are proposed in [5], [11]. In contrast to other multi-keyword search solutions that are based on either hashing or matrix multiplications, the results returning from bilinear pairing based solutions are free from false negatives and false positives (i.e., only the correct results return). However, computation costs of pairing based solutions are significantly high both on the server as well as on the user side. Our proposed work provides several orders of magnitude faster solution compared to [5], [11]. Moreover, those schemes do not provide any additional privacy for hiding access or search patterns of users. Therefore, pairing based solutions are not practical in many applications.

III. Preliminaries

The fundamental problem of privacy-preserving search is examining the similarity of items. We use a well known technique, known as minhashing [12] to deduce the similarity between sensitive data and the given encrypted query. We also utilize some of the metrics used in information systems to estimate the order of relevancy of the matching results. We present the definitions and the basics of these techniques in Sections III-A and III-B, respectively.

A. Minhashing

Each document is represented by a small set called signature. The important property of signatures is that, it should be possible to compare two signatures and estimate a distance between the underlying sets without any other information. Although the exact similarity cannot be deduced from the signatures, they still provide a good approximation. Moreover, the accuracy of the similarity further increases as larger signatures are used. The signatures are composed of several elements, each of which is constructed using *minhash* functions [12].

Definition 1: ***minhash*:** Let Δ be a finite set of elements, P be a permutation on Δ and $P[i]$ be the i^{th} element in the permutation P. *Minhash* of a set $D \subseteq \Delta$ under permutation P is defined as:

$$h_P(D) = min(\{i \mid 1 \leq i \leq |\Delta| \ \wedge \ P[i] \in D\}). \quad (1)$$

In the proposed method, for each signature, λ different random permutations on Δ are used so the final signature of a set D is:

$$Sig(D) = \{h_{P_1}(D), \ldots, h_{P_\lambda}(D)\}, \quad (2)$$

where h_{P_j} is the *minhash* function under permutation P_j.

B. Relevancy Score

In order to sort the matching results according to their relevancy to the query, a similarity function is required. This function assigns a relevancy score to each matching result corresponding to a given search query.

A commonly used weighting factor for information retrieval is tf-idf weighting [13]. Intuitively, it measures the importance of a search term within a document for a database collection.

The weight of each search term in each document is calculated using the tf-idf weighting scheme that assigns a composite weight using both term frequency (tf) and inverse document frequency (idf) informations. The tf-idf of a search term w in a document D is given by:

$$\text{tf-idf}_{w,D} = \text{tf}_{w,D} \times \text{idf}_w, \tag{3}$$

where tf is the number of times a keyword appears in a document and idf is the rarity of a search term within the database collection.

IV. Framework

In this paper, we are considering privacy-preserving keyword search over encrypted cloud data for the database outsourcing scenario. In this setting, we assume the data owner does not have sufficient resources or is unwilling to store the whole database. He outsources the data to an untrusted, semi-honest server, but maintains the ability to search without revealing anything except the access and search patterns. The data owner encrypts the sensitive documents to be outsourced and generates a secure searchable index using the features of these sensitive documents. In an offline stage, both searchable index and the encrypted documents are outsourced to a semi-honest cloud. Utilizing the searchable indexes, authorized users can perform search on the cloud and receive the encrypted documents that match with their queries. During this process, the cloud server should not learn anything other than what the data owner allows to leak. Finally, user decrypts the retrieved documents using the decryption key.

The method is formalized as follows. Let $\mathcal{D}$ be the set of sensitive documents and F_i be the set of features (i.e., keywords) of $D_i \in \mathcal{D}$. There are four algorithms in the scheme, namely: setup, index generation, query generation and search.

1) $Setup(\Psi)$: Given a security parameter Ψ, it generates a secret key $K \in \{0,1\}^{\Psi}$.
2) $IndexGeneration(K, \mathcal{D})$: Given the collection of sensitive documents $\mathcal{D}$, it extracts the feature set F_i for each document $D_i \in \mathcal{D}$ and generates a searchable secure index $\mathcal{I}$ via encryption with the key K.
3) $QueryGeneration(K, F)$: Generates a query Q for the given set of features F with key K.
4) $Search(\mathcal{I}, Q)$: Query Q is compared with the searchable index $\mathcal{I}$ and returns encrypted versions C_i of the matching documents D_i.

The details of these algorithms are given in Section V.

A. Security Model

The privacy definition for almost all of the existing efficient privacy-preserving search schemes allows the server to learn some information such as the search and access patterns. In case there is a need for hiding the access patterns, traditional private information retrieval (PIR) methods [14], [15] or Oblivious RAM [16] can be utilized for the document retrieval process. However these methods are not practical even for medium sized datasets due to incurred polylogarithmic overhead. Therefore, due to efficiency concerns, the proposed method also leaks similarity and access pattern.

Definition 2: **Search Pattern (S_p)** is the frequency of the queries searched, which is found by checking the equality between two queries. Formally, Let $\{Q_1, Q_2, \ldots, Q_n\}$ be a set of n consecutive queries, and F_i be the feature set of Q_i. Search pattern S_p is an $n \times n$ binary matrix, where $S_p(i,j) = 1 \Leftrightarrow Q_i = Q_j$.

Definition 3: **Similarity Pattern** (Sim_p) is same with S_p with the extension for multiple features. Let feature set of Q_i be $F_i = \{f_i^1, \ldots, f_i^y\}$ and $\left(\{f_1^1, \ldots, f_1^y\}, \ldots, \{f_n^1, \ldots, f_n^y\}\right)$ be the feature set of n queries. $Sim_p[i[j], p[r]] = 1$ if $f_i^j = f_p^r$ and 0 otherwise for $1 \leq i, p \leq n$ and $1 \leq j, r \leq y$.
Intuitively, similarity pattern is the information of common features between two queries.

Definition 4: **Access Patten (A_p)** is the collection of data identifiers that contains search results of a user query. Let F_i be the feature set of Q_i and $R(F_i)$ be the collection of identifiers of data elements that matches with feature set F_i, then $A_p = R(F_i)$.

Definition 5: **History (H_n)** Let $\mathcal{D}$ be the collection of documents in the dataset and $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$ be a collection of n consecutive queries. The n-query history is defined as $H_n(\mathcal{D}, \mathcal{Q})$.

Definition 6: **Trace ($\gamma(H_n)$)** Let $C = \{C_1, \ldots, C_l\}$ be the set of encrypted documents, $id(C_i)$ be the identifier of C_i and $|C_i|$ be the size of C_i. The trace of H_n is defined as $\gamma(H_n) = \{(id(C_1), \ldots, id(C_l)), (|C_1|, \ldots, |C_l|), Sim_p(H_n), A_p(H_n)\}$. We allow to leak the trace to an adversary and guarantee no other information is leaked.

Definition 7: **View (v)** is the information that is accessible to an adversary. Let $\mathcal{I}$ be the secure searchable index and, $id(C_i)$ and $\mathcal{Q}$ are as defined above. The view of H_n is defined as $v(H_n) = \{(id(C_1), \ldots, id(C_l)), C, \mathcal{I}, \mathcal{Q}\}$.

Definition 8: **Adaptive Semantic Security** [3] A cryptosystem is adaptive semantically secure, if for all probabilistic polynomial time algorithms (PPTA), there exists a simulator $\mathcal{S}$ such that, given the trace of a history H_n, $\mathcal{S}$ can simulate the view of H_n with probability $1 - \epsilon$, where ϵ is a negligible probability. Intuitively, all the information accessible to an adversary (i.e., view ($v(H_n)$)) can be constructed from the trace ($\gamma(H_n)$) that is allowed to leak.

V. Proposed Scheme

In this section, we provide the crucial steps of our proposed method. Search over encrypted cloud is performed through an encrypted searchable index that is generated by the data owner and outsourced to a cloud server. Given a query, server compares the query with the searchable index and returns the results without learning anything other than the information that is allowed to be leaked due to efficiency concerns.

A. Secure Index Generation

Our proposed method utilizes the idea of bucketization which is a data partitioning technique widely used in literature [17], [18], [4]. Here, each object is distributed into several

buckets via *minhash* functions introduced in III-A and the bucket-id is used as an identifier for each object in that bucket. This method maps objects such that the number of buckets, in which two objects collide, increases as the similarity between those objects increases. In other words, while two identical objects collide in all of the buckets, number of common buckets decreases as dissimilarity between objects increases. The proposed secure index is generated by the data owner utilizing the following phases, namely: feature extraction, bucket index construction and bucket index encryption.

1) **Feature Extraction:** For each document $D_i \in \mathcal{D}$, the set of features $F_i = \{f_{i1}, \ldots, f_{iz}\}$ that characterize the document is extracted. In our case, those features are composed of two values $f_{ij} = (w_{ij}, rs_{ij})$. The first one is a keyword w_{ij} of the sensitive document. The second one is the relevancy score (rs), which is based on tf-idf value of the keyword w_{ij} for document D_i as explained in Section III-B. This relevancy score is later used in the search method (cf. Section V-C) while ranking the matching results.

2) **Bucket Index Construction:** We first construct a *minhash* structure by selecting λ random permutations[1] on the set of all possible search terms (Δ). We then apply *minhash* on the first values of each feature set $F_i^* = \{w_{i1}, \ldots, w_{iz}\}$ as shown in Section III-A and generate a signature for each document as:

$$Sig(D_i) = \{h_{P_1}(F_i^*), \ldots, h_{P_\lambda}(F_i^*)\}. \quad (4)$$

Note that $\forall i \in \{1, \ldots, \lambda\},\ h_{P_i}(F_j^*) \in F_j^*$. In other words, each signature element of a document is a keyword for that document.

Then, feature set of each document is mapped to λ buckets using the elements of the signature of the document. Suppose $h_{P_i}(F_j^*) = w_k$, then we create a bucket with bucket identifier B_k^i, and identifiers and relevancy scores of all the documents that satisfy this property are added to this bucket. Bucket content is a vector of integer elements of size l where l is the number of documents in the outsourced dataset. Let B_k^i be a bucket identifier and $V_{B_k^i}$ be the integer vector, $V_{B_k^i}[id(D_j)] = rs_{jk}$ if and only if $h_{P_i}(F_j^*) = B_k^i$ and $V_{B_k^i}[id(D_j)] = 0$, otherwise.

3) **Bucket Index Encryption:** Bucket identifier B_k^i is a sensitive information since it may reveal a search term in a query that matches with a bucket, so it must be kept encrypted. Moreover, the server should be able to map the given encrypted bucket id to the one kept in the server without knowing the decryption keys. Hence, the encryption method used for hiding the bucket identifier must be a deterministic scheme. One of the most efficient methods that hides a value in a deterministic way is HMAC functions which are essentially cryptographic hash functions that utilize secret keys. In our proposed scheme, an HMAC function is used for hiding bucket identifiers. The secret key of HMAC function (K_{id}) is only known by the data owner and never revealed to the server. We denote the encrypted bucket identifier as $\pi_{B_k^j} = HMAC_{K_{id}}(B_k^j)$.

[1]The permutations are publicly shared with authorized users.

The content of a bucket ($V_{B_k^i}$) possesses sensitive information such as the pseudo identifiers of the documents in that bucket and their relevancy scores. These information must also be protected from the untrusted server, hence should be outsourced to the server only after encryption. A proper approach for encrypting bucket contents would be using a PCPA-secure (Pseudorandomness against chosen plaintext attacks) encryption method such as AES in CTR mode with a secret key ($K_{content}$). We denote the encrypted content vector as $\mathcal{V}_{B_k^j} = Enc_{K_{content}}(V_{B_k^j})$.

Let max be the maximum number of buckets that may occur in the index and cnt be the number of real buckets in the index. We add $max - cnt$ dummy elements to the index in order to hide the number buckets. The dummy elements $(\pi_{dum_i}, \mathcal{V}_{dum_i})$ are randomly generated with the condition that

$$|\pi_{B_k^j}| = |\pi_{dum_i}| \quad \text{and} \quad |\mathcal{V}_{B_k^j}| = |\mathcal{V}_{dum_i}|.$$

The secure index generation method is summarized in Algorithm 1.

Algorithm 1 Index Generation

Require: Δ:set of possible keywords, $\mathcal{D}$: collection of documents, h: λ *minhash* functions, Ψ: security parameter

```
K_id = Setup(Ψ), K_content = Setup(Ψ)
for all D_i ∈ D do
    F_i ← extract features of D_i
    Sig(D_i) = {h_{P_1}(F_i^*), ..., h_{P_λ}(F_i^*)}
    for j = 1 → λ do
        B_k^j = Sig(D_i)[j − 1]
        if B_k^j ∉ bucket identifier list then
            add B_k^j to bucket identifier list
            create V_{B_k^j}
        end if
        add rs_ik to vector V_{B_k^j}[id(D_i)]
    end for
end for
for all B_k^j ∈ bucket identifier list do
    π_{B_k^j} ← HMAC_{K_id}(B_k^j)
    V_{B_k^j} ← Enc_{K_content}(V_{B_k^j})
    add (π_{B_k^j}, V_{B_k^j}) to secure index I
end for
add max − cnt dummy elements (π_{dum_i}, V_{dum_i})
return I
```

Subsequent to the index generation, data owner encrypts each document in the dataset $\mathcal{D}$ as $\Omega_{id(D_i)} = Enc_{K_{data}}(D_i)$ and outsources this set of encrypted documents E_{Doc} to the server with the $\mathcal{I}$, where

$$E_{Doc} = \{(id(D_1), \Omega_{id(D_1)}), \ldots, (id(D_{|\mathcal{D}|}), \Omega_{id(D_{|\mathcal{D}|})})\}.$$

B. Query Generation

The query generation is constructed in a similar way to the index generation phase (Section V-A).Given a feature set of n

keywords to be queried (i.e., $F = \{w'_1, \ldots, w'_n\}$), the user first creates the query signature from this feature set using the same *minhash* functions that are used in index generation phase. Then, for each signature element, the corresponding bucket identifier is hashed with the key K_{id}. The query Q is this list of hashed bucket identifiers (i.e., $Q = \{\pi_1, \ldots, \pi_\lambda\}$). Note that independent of the number of keywords in a query (n), the query signature has λ elements and therefore, the information of n is not leaked to the server.

C. Secure Search

Given a query Q, server finds the encrypted vectors ($\mathcal{V}_{B_k^j}$) corresponding to the bucket identifiers in Q. The server then sends back the λ encrypted vectors $E_V = \{\mathcal{V}_1, \ldots, \mathcal{V}_\lambda\}$ to the user. After receiving the buckets, user decrypts the vectors and ranks the data identifiers as it is detailed in Section V-D.

D. Document Retrieval

The user wants to avoid returning unrelated documents since this immediately bring forth an unnecessary communication burden. Hence, user tends to retrieve only the top t matches, instead of returning all documents that share at least one bucket with the query. The standard formulation for calculating the document-term weights is tf-idf [19] which is commonly used for relevance score calculation in search methods. Therefore, we also utilize tf-idf values for ranking the matching results.

Upon receiving the encrypted vectors $E_V = \{\mathcal{V}_1, \ldots, \mathcal{V}_\lambda\}$, the user decrypts those vectors and get the plain vectors as $V_i = Dec_{K_{content}}(\mathcal{V}_i)$. Then the documents are sorted according to their scores. Note that $V_i[id(D_j)]$ is the tf-idf value of document D_j for i^{th} bucket.

In the index generation phase each document is mapped to certain number of buckets using the output of *minhash* functions and tf-idf value of the *minhash* output is assigned as the relevancy score of that document for that bucket. Similarly query Q is also mapped to some buckets. The score of a document D_j (i.e., $score(id(D_j))$) is the summation of the relevancy scores for the buckets that both document and query share, which is defined as follows:

$$score(id(D_j)) = \sum_{i=1}^{\lambda} V_i[id(D_j)]. \tag{5}$$

As the $score(id(D_j))$ gets higher, the relevancy of the document to the query is expected to increase.

After the ranking phase, user retrieves the top t matches from the server. The document retrieval method is summarized in Algorithm 2. Note that as database is updated by adding or removing documents, tf-idf values need to be recalculated and indices should be updated accordingly. However, we assume the database is highly static, hence update is done infrequently.

Algorithm 2 Document Retrieval

USER:

Require: E_V: encrypted vectors, $K_{content}$: secret key, t: limit for number of documents to retrieve

for all $\mathcal{V}_i \in E_V$ **do**
  $V_i \leftarrow Dec_{K_{content}}(\mathcal{V}_i)$
end for
for $j = 1 \rightarrow |V_i|$ **do**
  $score(j) = \sum_{i=1}^{\lambda} V_i[j]$
end for
sort $score$ list
idList $\leftarrow$ identifiers of top t $scores$
send idList to Server

SERVER

Require: idList: requested document identifiers, E_{Doc}: outsourced encrypted documents

for all $id \in$idList **do**
  if $(id, \Omega_{id}) \in E_{Doc}$ **then**
    send (id, Ω_{id}) to user
  end if
end for

USER:
$D_{id} \leftarrow Dec_{K_{data}}(\Omega_{id})$

VI. Privacy

The privacy-preserving search scheme that we propose is adaptive semantically secure according to Definition 8.

Theorem 1: The proposed method satisfies adaptive semantic security in accordance with Definition 8.

Proof:

Let the original view $v(H_n)$ and the trace $\gamma(H_n)$ be

$$v(H_n)=\{(id(C_1), \ldots, id(C_l)), C, \mathcal{I}, \mathcal{Q}\},$$
$$\gamma(H_n)=\{(id(C_1), .., id(C_l)), (|C_1|, .., |C_l|), Sim_p(H_n), A_p(H_n)\}.$$

Further let $v^*(H_n) = \{(id^*(C_1), \ldots, id^*(C_l)), C^*, \mathcal{I}^*, \mathcal{Q}^*\}$ be the view simulated by the simulator S. The proposed method is adaptive semantically secure if $v(H_n)$ is indistinguishable from $v^*(H_n)$.

- The first component of the view $view(H_n)$ is the document identifiers $id(C_i)$ which are also available in trace. Hence, S can trivially simulate document identifiers as $id^*(C) = id(C)$. Since $id^*(C) = id(C)$, they are indistinguishable.
- Each document is encrypted using a PCPA-secure encryption method (e.g., AES in CTR mode). The output of a PCPA-secure encryption method [3] is by definition indistinguishable from a random number that has the same size with ciphertext. To simulate ciphertexts C, S assigns l random numbers to C^* such that $C^* = \{C_1^*, \ldots, C_l^*\}$ where $\forall i, |C_i^*| = |C_i|$. Note that size of each ciphertext is available in the trace. Considering for all i, C_i and C_i^* are indistinguishable, C and C^* are also indistinguishable.
- Note that $\mathcal{I}$ is composed of encrypted bucket identifiers and corresponding encrypted bucket content vectors. Let $size_B$ and $size_V$ be the sizes of bucket identifier and

bucket content, respectively. Further let max be the maximum number of buckets that may occur in $\mathcal{I}$. Simulator S generates max index elements, $\mathcal{I}^*[i] = (\pi_i^*, \mathcal{V}_i^*)$ such that π_i^* is a random number, where $|\pi_i^*| = size_B$ and $\mathcal{V}_i^*$ is another random number, where $|\mathcal{V}_i^*| = size_V$. Note that π_i^* and π_i are indistinguishable since π_i is the output of a random function (i.e., HMAC) where the output is indistinguishable from a random number. Similarly, $\mathcal{V}_i^*$ and $\mathcal{V}_i$ are indistinguishable since $\mathcal{V}_i$ is a cipher of a PCPA-secure encryption method. Hence, $\mathcal{I}$ is indistinguishable from $\mathcal{I}^*$.

- $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$ is a set of n consecutive queries where each query Q_i is composed of λ encrypted bucket identifiers (i.e., $Q = \{\pi_1, \ldots, \pi_\lambda\}$). S can simulate the queries using the similarity pattern (Sim_p). Let $Q_i[j]$ be the j^{th} element of Q_i where $size_B$ is the size of bucket identifier. If $\exists p, r\ 1 \leq p \leq i$ and $1 \leq r \leq \lambda$ such that $Sim_p[i[j], p[r]] = 1$ set $Q_i^*[j] = Q_p^*[r]$. Otherwise, set $Q_i^*[j]$ to a random value R_i^j where $|R_i^j| = size_B$. Note that for all i, Q_i is indistinguishable from Q_i^* since Q_i is the output of a pseudorandom permutation and Q_i^* is a random number, and they are of the same length.

The simulated view v^* is indistinguishable from genuine view v since each component of v and v^* are indistinguishable. Hence, the proposed method satisfies adaptive semantic security. ■

VII. Two Server Search

In the proposed scheme, it is possible to correlate an encrypted query with document identifiers of corresponding matching documents which is also the case for most of the privacy-preserving search schemes with the exception of Oblivious RAM based solutions. In order to prevent such a correlation, we introduce a second server, referred as file server, that do not collude with the initial server, which is referred as search server henceforth. While the search server returns the encrypted vectors for a given query, the encrypted documents are retrieved from the file server. With this approach, the search server does not learn document identifiers of the retrieved documents and the file server does not learn the query. Therefore, assuming the two servers do not collude with each other, correlating a query with the corresponding document identifiers is not possible.

With the existence of two servers, they can also be utilized to perform some work on behalf of the user. The document retrieval phase explained in Section V-D can be a heavy burden on user depending on the user's capabilities. The user should calculate and then sort the scores of all document identifiers subsequent to decrypting the retrieved encrypted vectors. Unlike to a server, users may be using resource-constraint devices. In order to relieve the burden of the user, the file server can be utilized to perform sorting the scores of the matching document identifiers.

In the two server approach, the relevancy score of each document identifier is calculated by the search server. However, due to the privacy requirements, the search server should learn neither the individual scores nor the order between the scores. This implies that decryption of the encrypted vectors by the search server should not be possible. The homomorphic encryption schemes enable computation over encrypted values which are appropriate for our case. We use the Paillier encryption [20], a well known additive homomorphic encryption method, in the encryption of the bucket content vectors $V_{B_k^i}$. The Paillier encryption satisfies the property that, $Dec(Enc(m_1, r_1) \cdot Enc(m_2, r_2)) = m_1 + m_2$, where the search server utilizes to compute $Enc(score(j)) = \sum_{i=1}^{\lambda} Enc(V_i[j])$ for each document identifier.

The file server gets the matching Paillier encrypted bucket content vectors and decrypts the results. Then the plain scores are sorted and matching items with top t relevancy scores are sent to the user. With this approach all the computation burden of the user is transferred to the servers at the cost of increasing the size of encrypted bucket content vectors. In the single server approach each element of the vector is a 32 bit integer, while in the two-server approach, each element is a $\lceil \log_2 n^2 \rceil$-bit ciphertext, where n is multiple of two large prime numbers. Nevertheless, this vector is only transferred between the two servers which are known to possess vast resources of computation and communication; hence the technique does not affect the communication cost of the user.

The two-server search method is described in Algorithm 3.

Algorithm 3 Two-Server Secure Search and Document Retrieval

SEARCH SERVER:

Require: $\mathcal{I}$: secure index, Q: query, n Paillier modulus, t: limit for number of documents to retrieve

for all $\pi_i \in Q$ **do**
  if $(\pi_i, \{e_{i_1}, \ldots, e_{i_l}\}) \in \mathcal{I}$ **then**
    $Enc(score(j)) \leftarrow Enc(score(j)) \cdot e_{i_j}$
  end if
end for
send $(j, Enc(score(j)))$ and t to File Server

FILE SERVER:

Require: $K_{content}$: secret key, K_{priv} : Paillier private key

for all i **do**
  $score(i) = Dec_{K_{priv}(Enc(score(i)))}$
end for
sort all scores
send encrypted documents corresponding to the highest t scores

VIII. Experiments

In this section, we extensively analyze the proposed method in order to demonstrate the efficiency and effectiveness of the scheme. The entire system is implemented by Java language using a 32-bit Windows 7 operating system with Intel Pentium Dual-Core processor of 2.30GHz. In our experiments we use the publicly available Enron dataset [21].

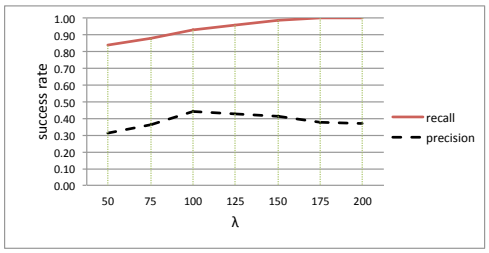


Fig. 1: Success Rates as λ change for $t = 15$

The success of a search scheme can best be analyzed using the precision and recall metrics. Let $R(F)$ be the set of items retrieved for a query with feature set F and $R^*(F)$ be a subset of $R(F)$ such that, the elements of $R^*(F)$ include all the features in F. Further let $D(F)$ be the set of items that contain all the features in F. Note that $R^*(F) \subseteq R(F)$ and $R^*(F) \subseteq D(F)$. Precision ($prec(F)$), recall ($rec(F)$), average precision ($aprec(F)$) and average recall ($arec(F)$) for a set $\mathcal{F} = \{F_1, \ldots, F_n\}$ are defined as follows:

$$prec(F) = \frac{R^*(F)}{R(F)}, \quad aprec(\mathcal{F}) = \sum_{i=1}^{n} \frac{prec(F_i)}{n} \tag{6}$$

$$rec(F) = \frac{R^*(F)}{D(F)}, \quad arec(\mathcal{F}) = \sum_{i=1}^{n} \frac{rec(F_i)}{n} \tag{7}$$

The matching items are ordered according to the relevancy scores (cf. Section III-B) and only items with top t scores are retrieved. We analyzed the effect of the number of *minhash* functions (λ) on the accuracy of the method for a fixed threshold $t = 15$, by taking the average of 1500 queries with number of features differ from 2 to 6 (i.e., 300 queries per each feature size). As Figure 1 demonstrates, recall of the proposed scheme is 1 for any $\lambda \geq 150$ implying that all of the items that contain all the features in the given query are retrieved by the user. For the database outsourcing scenario that we consider, it is crucial that the user retrieves all the documents matching with the queried feature set. Precision is rather small, which indicates about 40% of the retrieved documents contain all the queried features. Nevertheless, the other retrieved items are still relevant with the query. Those items contain a subset of the query features and the matching features have high relevancy scores indicating that the matching item is highly relevant to the query even when not all the features are captured. Note that, an item that has no matching feature with a query has zero relevancy score, hence cannot match with the query. We set $\lambda = 150$ since it satisfies the best precision rate while ensuring full recall.

We analyze the impact of the number of keywords in a query on the precision and recall rates and present the results in Figure 2. The similarity between query and document signatures increases as the number of common keywords increases. Hence, both the precision and recall rates of the method increase as the number of keywords in a query increases. The increase in success rate indicates our proposed method is even more useful for searches with more than 5 keywords.

We test the efficiency of our proposed method using various dataset sizes from 4000 to 10000 documents. The most costly operation of our method is index generation. Figure 3 shows that the index generation operation takes about a few minutes and linearly increases as the number of documents increases. Considering this operation is only performed in an offline stage by the data owner, the method is practical. One of the most important parameters of privacy-preserving search is query response time since this operation is used very frequently and users want to access their search results as fast as possible. Search operation does not depend on the number of documents since, in the proposed method search is performed by retrieving λ requested buckets which is constant. This feature is especially important for huge datasets where the number of documents is in the order of millions. The average query response time for our method for $\lambda = 150$ is 210 ms independent of the number of documents in the dataset.

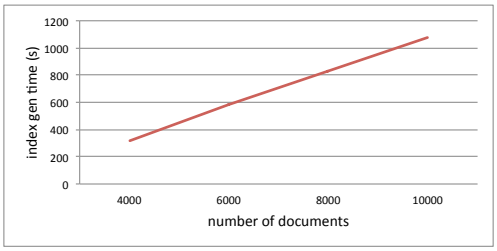
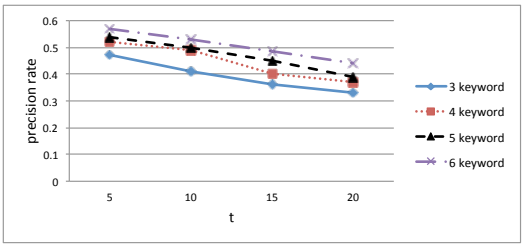


Fig. 3: Timings for index construction for $\lambda = 150$

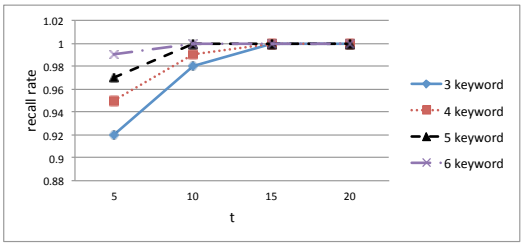
If two-server search method is utilized, the file server needs to decrypt the Paillier encrypted scores of each document in dataset $\mathcal{D}$. A single Paillier decryption operation using 1024-bit primes, takes about 90 ms in the computer that we used in our experiments. Therefore, two-server setting has about $90 \cdot |\mathcal{D}|$ ms additional cost on the search due to decryption. Nevertheless, decryption operation can be highly parallelized and by utilizing high performance computers on the file server, actual cost of decryption can significantly be reduced.

The communication cost of the user for the single server case has two phases. First, the encrypted matching vectors ($|E_V| = \lambda|\mathcal{V}_i|$bits) are received and in the next phase matching encrypted documents are received. However, using two server setting, only the matching encrypted documents are sent to the user which is the minimum communication possible.

Most of the secure search methods in literature do not support multiple features in queries. We do not provide any comparison with those single keyword search methods but compare our proposed method with the existing secure multi-keyword search methods instead. Some of the multi-keyword search methods utilize bilinear mapping such as [5]. This approach has similar security requirements with our proposed method, such that it reveals search and access pattern but nothing else. In this work, each search operation does about $2l$ bilinear mapping operation where l is the number of features in a document, which is not practical due to the cost of bilinear map operations. A recent work by Cao et al. [7] utilizes matrix multiplication operations where the number of rows is determined by the size of the complete feature set. This method performs index construction for 6000 documents in



(a) precision



(b) recall

Fig. 2: Impact of number of keywords in a query and t on the precision (a) and recall (b) rates

about 4500 s while we perform the same operation in less than 600 s. Similarly, the search operation over 6000 documents in [7] requires 600 ms, while we perform in about 210 ms. Moreover, our search time is independent from the number of documents so our method has paramount advantage as the number of documents goes in the order of hundreds of thousands. Another multi-keyword search method is proposed by Orencik and Savas [6]. This work performs efficiently in both index construction and search operations. Similar to [7], the search time of [6] is also linear in the number of documents, therefore, our proposed method performs better in search for large datasets.

IX. Conclusion

In this work, we addressed the privacy-preserving multi-keyword search over encrypted cloud data for the database outsourcing scenario. We present a novel method using *min-hash* functions that provide efficient comparison between signatures of documents and queries. We provide formal security definitions and prove that our proposed work satisfies adaptive semantic security. We incorporate ranking capability to the proposed scheme utilizing well known tf-idf based relevancy scoring. This approach ensures that only the most relevant items are retrieved by the user, preventing unnecessary communication and computation burden on the user. We implement the entire system and demonstrate the effectiveness and efficiency of our solution through extensive experiments using the publicly available Enron dataset [21].

Acknowledgment

Cengiz Orencik was supported by the Ph.D. fellowship of TUBITAK (The Scientific and Technological Research Council of Turkey). Dr. Kantarcioglu was partially supported by Air Force Office of Scientific Research MURI Grants FA9550-08-1-0265 and FA9550-12-1-0082, National Science Foundation (NSF) Grants Career-CNS-0845803, CNS-0964350, CNS-1016343, CNS-1111529, CNS-1228198. Dr. Savas was partially supported by Turk Telekom under Grant Number 3014-07.

References

[1] W. Ogata and K. Kurosawa, "Oblivious keyword search," in *Journal of Complexity, Vol.20*, 2004, pp. 356–371.

[2] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data." in *ICDCS'10*, 2010, pp. 253–262.

[3] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proceedings of the 13th ACM conference on Computer and communications security*, ser. CCS '06, 2006, pp. 79–88.

[4] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ser. ICDE '12, 2012, pp. 1156–1167.

[5] B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 262–267, Jan. 2011.

[6] C. Örencik and E. Savaş, "Efficient and secure ranked multi-keyword search on encrypted cloud data," in *Proceedings of the 2012 Joint EDBT/ICDT Workshops*. ACM, 2012, pp. 186–195.

[7] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *IEEE INFOCOM*, 2011.

[8] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, Report 2003/216, 2003.

[9] M. Raykova, B. Vo, S. M. Bellovin, and T. Malkin, "Secure anonymous database search," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, ser. CCSW '09. ACM, 2009, pp. 115–126.

[10] P. Wang, H. Wang, and J. Pieprzyk, "An efficient scheme of common secure indices for conjunctive keyword-based retrieval on encrypted data," in *Information Security Applications*, ser. Lecture Notes in Computer Science. Springer, 2009, pp. 145–159.

[11] Z. Chen, C. Wu, D. Wang, and S. Li, "Conjunctive keywords searchable encryption with efficient pairing, constant ciphertext and short trapdoor," in *PAISI*, 2012, pp. 176–189.

[12] A. Rajaraman and D. Ullman, Jeffrey, *Mining of massive datasets*. Cambridge University Press, 2011.

[13] H. S. Christopher D. Manning, Prabhakar Raghavan, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[14] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. Skeith, "Public key encryption that allows pir queries," in *Advances in Cryptology - CRYPTO 2007*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007, vol. 4622, pp. 50–67.

[15] H. Lipmaa, "First cpir protocol with data-dependent computation," in *Information, Security and Cryptology - ICISC 2009*. Springer, 2009, pp. 193–210.

[16] B. Pinkas and T. Reinman, "Oblivious ram revisited," in *Proceedings of the 30th annual conference on Advances in cryptology*, ser. CRYPTO'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 502–519.

[17] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing sql over encrypted data in the database-service-provider model," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '02, 2002, pp. 216–227.

[18] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *The VLDB Journal*, vol. 21, no. 3, pp. 333–358, Jun. 2012.

[19] J. Zobel and A. Moffat, "Exploring the similarity space," *SIGIR FORUM*, vol. 32, pp. 18–34, 1998.

[20] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *ADVANCES IN CRYPTOLOGY - EUROCRYPT 1999*. Springer-Verlag, 1999, pp. 223–238.

[21] "Enron email dataset," http://www.cs.cmu.edu/enron, Jan. 2012.