The 7th International Symposium on Emerging Information, Communication and Networks (EICN 2020) 
November 2-5, 2020, Madeira, Portugal

An Effective and Efficient Technique for Supporting Privacy-Preserving Keyword-Based Search over Encrypted Data in Clouds

Alfredo Cuzzocrea\textsuperscript{a,*}, Carson K. Leung\textsuperscript{b}, Bryan H. Wodi\textsuperscript{b}, S. Sourav\textsuperscript{b}, Edoardo Fadda\textsuperscript{c,d}

\textsuperscript{a}University of Calabria, Rende, Italy and LORIA, Nancy, France
\textsuperscript{b}University of Manitoba, Winnipeg, MB, Canada
\textsuperscript{c}Politecnico di Torino, Torino, Italy
\textsuperscript{d}ISIRES, Torino, Italy

Abstract

Nowadays, cloud providers offer to their clients the possibility of storage of emails and files on the cloud server. To avoid privacy concerns, encryption should be applied to data. Unlike searching plaintext documents by keywords, encrypted documents cannot be retrieved in the same manner. As keyword searches on encrypted data are in demand, this paper describes an effective and efficient technique to support privacy-preserving keyword-based search over encrypted outsourced data. With this technique, encrypted data are first searched with the keyword, support for dynamic operations is then checked, and all relevant data documents are finally sorted based on the number of keywords matching the user query. To evaluate the technique, precision and recall are measured. The results reveal the effectiveness and efficiency of the technique in supporting privacy-preserving keyword-based search over encrypted outsourced data.

© 2020 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the Conference Program Chairs.

Keywords: emerging systems; emerging information; privacy preserving search; information retrieval; encrypted data; outsourced data

1. Introduction

Data are of high value in the present and future decades. Nowadays, it is possible to gather, and analyze huge amount of data from different sources. Valuable knowledge and information are embedded in these big data [13, 40]. Hence, a field of interest is data science [31]. In order to learn [14, 26, 34], mine [28, 35], retrieve from [10, 33, 44, 50],
and visualize [1, 30, 37] big data, it is common to consider different aspects (e.g., 3V’s, . . . , 7V’s) [9] of big data. The main goal of data analysis is to retrieve information and discover knowledge, via techniques like data mining. However, privacy concerns can limit the distribution and analysis of the data [39]. Different business analytics or business intelligence (BI) solutions [2, 21]—in terms of data analysis, mashups and services—can be brought if anonymity could be guaranteed. As much data are related to personal information (especially, social network data [3, 22, 36, 47]), privacy is of central importance, like also highlighted in similar studies (e.g., [53]).

Nowadays, a very large volume of data involving sensitive information about an individuals and organizations are generated by data owners by means of various applications. Technologies such as hard disk, DVDs, floppy disk and many others enable data owners to store these data on local machine. Storing data in, and retrieving data from, local machines requires the management and integrity of data, and incurs long time and expensive hardware [51]. The possibility of storing data on cloud infrastructure without the purchase of computational resources brought by advances in cloud computing as it reduces the operational overhead and let the use to access ubiquitous services [42].

Despite its advantages, privacy concerns may arise if sensitive information regarding an organization or an individual are loaded on the cloud service. For this reason, cloud service providers (CSPs) are providing privacy-preserving solutions [16, 41]. Nevertheless, these techniques are prone to attacks from malicious inside attackers who abuse the information without proper authorization. Furthermore, data confidentiality is compromised by outside attacks. Encrypting metadata of the owners prior to the outsource of data onto the cloud server is a simple approach to solve the confidentiality problem [24]. However, this approach makes it difficult to search, update and delete data. To overcome this new problem, a dummy strategy can be downloading all the encrypted data from the cloud server and then locally decrypting them to compute different tasks. However, this strategy is its high demand on the network bandwidth and its high hardware cost. Thus, being able to use multiple keywords to search encrypted data without lose of data confidentiality is really important. In particular, a goal is to design a search mechanism to retrieve data and rank the resulting documents—satisfying user input—according to their relevance in a cloud server based on a given input query. Another goal is to design a feature of providing most relevant data documents in sorted order from all searched documents to data consumers.

Key contributions of this paper include the theoretical design and practical implementation of a scheme for supporting fast secure keyword-ranked search on data collection in the cloud. The resulting scheme enables data owners for the insertion and deletion of their encrypted data to and from the cloud, while obtaining high search efficiency. The extension of a secure searchable encryption scheme also allows data owners the update of encrypted data on the cloud. The implementation of a parallel search process on encrypted data further reduces the time cost.

2. Background and related work

For CSPs, security is of paramount importance. Their responsibility is to allow the users to process, format, and transmit data to a remote location, as well as to protect the data from internal or external threats [48]. As unauthorized access to the data is a common problem in the public cloud, CSPs provide solutions like encryption, access control, virtualization to data owners to disseminate data [23]. One of the problem is crucial in the cloud environment is the secure keyword matching problem, which has gained attention. The existing methodologies for solving this problem include single-keyword searchable encryption [38, 45], Boolean keyword searchable encryption [25], and multi-keyword ranked searchable encryption. Single keyword techniques uses a symmetric encryption to store encrypted data documents and their encrypted searchable index on the server, and performs the searches with the trapdoor (TD) built from the data owner’s secret key. A techniques called index blind storage (IBS) mechanism [19] enhances the search security, enables an execution of range queries on encrypted cloud data and conceals the query pattern at the same time. To achieve blind indexing, the IBS stores a key hash of the plaintext (i.e., unencrypted information) in a separate column on the table, and generates an index on that column. Through this separation of encryption key and index key, the key cannot be obtained by the database server.

The cosine and term frequency (TF) × inverse document frequency (IDF) have been used in multi-keyword ranked searchable encryption schemes [46] for ranking similarity. However, while many of these methodologies provide secure keyword searchable encryption, update and delete operation on a given dataset on the cloud server are not possible. This justifies the need for the searchable encryption technique that supports dynamic operations.
3. Our privacy-preserving keyword search model

3.1. Architecture design

The architecture of our privacy-preserving keyword search (PPKS) model consists of three main components:

- **Data owners**, who aim to use the encrypted files for the search operation: They first build an encrypted tree index \( I \) for all the keywords gathered from collection \( F \) of raw documents belonging to the data owner who wishes to store at remote cloud location in the encrypted format. They then generate ciphertext (i.e., encrypted text transformed from plaintext) collection \( C \) for \( F \), outsource both \( C \) and \( I \) to the cloud storage server, and update the documents in \( F \) in the cloud server by encrypting them in local and load it to the server later on.

- **Cloud service provider (CSP)**, who stores both \( C \) and \( I \) in the remote location: Once load the file, the CSP (i) receives the search request—in the form of TD—from the data user, (ii) executes the search operation on \( I \) based on TD, and (iii) returns to the data user those relevant encrypted documents ranked according to the order \( k \). If the data owner loads new documents, the CSP performs update operation.

- **Data users**, who obtained the authorization from the data owner to possess a secret key to utilize encrypted data: They use the search control mechanism to either use \( t \) keywords with TD for handling the search query, or retrieve \( k \) encrypted documents from the cloud storage server based on the keyword in the query. An “honest-but-curious” CSP performs all the operations in an honest manner but tries to infer the knowledge from query data. For instance, it analyzes the inferred knowledge to gain additional information and perform various operations.

To summarize, data owners possess \( F \) and outsource both \( C \) and \( I \) to the CSP, who returns ranked results upon receiving search requests from data users, who in turn communicate with the data owners by using symmetric keys and search control via \( TD \).

Based on the information acquired by the cloud server, two key threat models include known ciphertext and background models. In a known ciphertext model, as the CSP has knowledge on both \( C \) and \( I \) outsourcing to the cloud by the data owner, ciphertext attack can be performed. In contrast, in a known background model, the CSP has knowledge on \( C \), term frequency (TF), as well as statistics about the number of documents for each keyword in the whole documents. So, the CSP tries to deduce the query keyword based on keyword frequency.

3.2. Properties

Based on the aforementioned design, our PPKS model enables security, accuracy, and dynamic keyword ranked search over encrypted data stored in cloud server. To elaborate, our PPKS model preserves privacy by making the CPS unable to learn additional information about the stored documents and only able to learn about search result. Consequently, it achieves:

- **index and query confidentiality**, by encrypting both \( F \) and keywords in the index and query so that the adversary is totally unaware of the data.
- **keyword privacy**, by hiding the underlying keyword used for the search. As TD does not prevent the adversary to make statistical inference for prediction of the keyword of the search query, background information about known ciphertext model can be used to find out the keyword of the search query.
- **trapdoor unlinkability**, by generating TD non-deterministic to avoid the possibility of deducing whether or not the two TDs are generated for the same query keyword. Through the generation of frequencies of keywords used in different search queries, the CSP is able to breach the privacy requirement of the underlying keywords in the search query. By using a non-deterministic approach to generate TD, it ensures the privacy requirement for the same keyword utilizing for the same search query.

In addition, our PPKS model ensures efficient query mechanism by following index based tree that offers sub-linear efficiency while searching. Moreover, it also handles the keyword queries on encrypted data stored in the cloud server and dynamic document updates on the cloud server.
3.3. Practical implementation

The designed architecture of our PPKS model with the aforementioned properties is implemented by a literal search of encrypted data for single keywords and then an expansion of its utility to perform partial match searching. Afterwards, it is further extended to accommodate multiple keywords by using the blind indexing strategy (e.g., IBS). As an example of the IBS, see the schema in Table 1. Here, the IBS stores a key hash of the plaintext in a separate column on the table, and generates an index on that column. After storing a blind index of the plain keywords in a separate column, the IBS encrypts the keyword and stores the results in the table. In general, our implemented PPKS model adapts a password-based key derivation function (PBKDF) 2 to reduce vulnerabilities to brute force attacks. It uses a hash-based message authentication code (HMAC), which involves the cryptographic secure hash algorithm 1 (SHA1) and a secret cryptographic key for data integrity and message authenticity. In other words, the PPKS model combines the hashing and a unique “salt” (i.e., random data) for deriving a key from the keyword. Hence, given a list of documents with keyword, the PPKS model builds a keyword blind index, searches for the index, and returns a list of documents containing the keyword.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Nullable?</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>integer</td>
<td>N</td>
<td>auto increment, primary key, referenced by filter ID in Table 2</td>
</tr>
<tr>
<td>UNIX name</td>
<td>text</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>file name</td>
<td>character string</td>
<td>N</td>
<td>unique</td>
</tr>
<tr>
<td>file size</td>
<td>integer</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>file path</td>
<td>character string</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>keyword</td>
<td>text</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>index value</td>
<td>character string</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>keyword blind index</td>
<td>character string</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

To further enhance the searching process for privacy-preserving information retrieval, our implemented PPKS model accommodates partial string matches. The creation of an associated index speeds up the search process. Continue with the earlier example, see the schema in Table 2 for the associated index on both the string match and TD.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Nullable?</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter ID</td>
<td>integer</td>
<td>N</td>
<td>auto increment, primary key</td>
</tr>
<tr>
<td>file upload ID</td>
<td>integer</td>
<td>N</td>
<td>referencing ID in Table 1</td>
</tr>
<tr>
<td>string match</td>
<td>text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TD</td>
<td>text</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Finally, for every type of the needed query, a distinct blind index per column is stored. To handle single keywords, a blind index of the first five characters for every keyword the user enters are store. It is different when handling multiple keywords.

4. Experimental assessment

The encryption key and index key are recommended to be store separately on the web server of the application because this would prevent the database server from obtaining the keys. In an event that unauthorized users access the database, they would be able to discover any private data besides the access and search patterns. This method solves the problem of insider as well as outsider attack. Hence, duplicate entries for the possible partial search strings are added to our table. By doing so, we aid indexing, which in turns enables fast SELECT queries. The creation of
an index for partial SELECT queries to match the first five letters reduces memory consumption. Furthermore, we truncate the blind indexes to 16 bits and use them as a Bloom filter to further boost the method performance in term of speed.

Moreover, we compare the performance of our PPKS model with related work. For example, Cao et al. [7] developed a multi-keyword search scheme, but their scheme is not dynamic so that data owners cannot update their data once uploaded to the cloud. To allow update, Sun et al. [46] regenerated indexes whenever data were updated, which can be computationally expensive. In contrast, our PPKS model stores a separate distinct blind index per column for each kind of query with its own key.

For the assessments, we computed the search times of multi-word queries on a collection of 2,000 unencrypted documents. When fed with the contents of these documents, we use a random word generator to generate keywords from a collection of five words, and use a text analyzer to find the most frequent phrases and frequencies of words. Search times, as well as percentages of RAM and thread loads, for queries with vs. without parallel scheme measured on a 32 GB Intel Core i7-7700HQ CPU with a 2.80 GHz processor, 8 threads, and a GeForce GTX 1060 graphics card. Here, the selected CSP is Dropbox, and API calls were used for uploading and retrieving files. Comparison results are shown in Table 3 and Fig. 1. As shown in the table, queries running in parallel required an average search time of less than 5 seconds, but the average search time grows proportional to the number of files when the search process was not executed in parallel. In the figure, memory usage is colored in red and average thread load is colored in green. It shows that the average thread load dropped when processing searches in parallel.

Table 3. Search time (a) with vs. (b) without parallel search process.

<table>
<thead>
<tr>
<th># files</th>
<th>(a) Search time (in seconds) with parallel search process</th>
<th>(b) Search time (in seconds) without parallel search process</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>1</td>
<td>9.4</td>
<td>0.9</td>
</tr>
<tr>
<td>5</td>
<td>5.2</td>
<td>0.4</td>
</tr>
<tr>
<td>10</td>
<td>4.7</td>
<td>0.2</td>
</tr>
<tr>
<td>50</td>
<td>4.2</td>
<td>0.4</td>
</tr>
<tr>
<td>100</td>
<td>4.5</td>
<td>0.3</td>
</tr>
<tr>
<td>200</td>
<td>5.1</td>
<td>0.7</td>
</tr>
<tr>
<td>500</td>
<td>5.2</td>
<td>0.7</td>
</tr>
<tr>
<td>1000</td>
<td>5.3</td>
<td>0.2</td>
</tr>
<tr>
<td>1500</td>
<td>5.1</td>
<td>0.5</td>
</tr>
<tr>
<td>2000</td>
<td>5.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Fig. 1. Memory utilization and thread load (a) without parallel search process vs. (b) with parallel search process.
5. Conclusion and future work

In this paper, we described an effective and efficient technique for supporting privacy-preserving keyword-based search (PPKS) over encrypted outsourced data. This well-designed and implemented PPKS model allows the data owner to perform dynamic search on encrypted data that are stored in the cloud. It implements a document ranking feature to support queries for encrypted keyword frequency, and maintains data privacy and integrity via the symmetric encryption/decryption scheme to transfer and query the encrypted contents in the cloud. Moreover, the model speeds up the search by a parallel search scheme. By using the symmetric key encryption, it guarantees the invulnerability to many types of attacks (except the physical access to the key which is almost impossible in the ideal world).

Future work will (i) exploit and incorporate user cognitive constraints [15, 27, 29] into the search, as well as (ii) transfer the learned knowledge—such as information from online analytical processing (OLAP) data [5, 6, 8, 11, 52], graphs [4, 12, 17, 20, 43, 49], web [18, 32, 54]—to effective and efficient PPKS models for supporting privacy-preserving keyword-based search over encrypted outsourced data.

Acknowledgements. This project is partially supported by NSERC (Canada) and University of Manitoba.

References


