A Lightweight Privacy Preserving Keyword Search Over Encrypted Data in Cloud Computing

Ibrahim El-henawy [1], Salwa H. Mahmoud [2,*] and Ahmed Moustafa [3]

[1] Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt; henawy2000@yahoo.com
[2] Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt; salwa_hassan_82@yahoo.com
[3] Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt; ahmad@zu.ed.eg

* Correspondence: salwa_hassan_82@yahoo.com

Abstract

With the emerging development of cloud computing services, data owners outsource their documents to Cloud Service Providers (CSP) which could lead to threats related to security and privacy. Hence, protecting the privacy of user data and providing queries privacy becomes one of the main concerns of the data owner. One of the solutions for providing privacy and confidentiality of the outsourced data is encrypting it before sending it to the cloud. Although this solution satisfies data confidentiality and prevents the CSP from reading or modifying the data without the data owner's permission, it prevents the data owner to search the outsourced documents directly. Symmetric encryption algorithms e.g. AES have a searching limitation, in which the whole encrypted document needs to be retrieved from the CSP and then decrypt before performing the search procedure. To overcome this limitation, a lot of keyword-based search approaches have been done. These approaches allow users to retrieve just those documents contain special keywords. However, most of these approaches suffer from privacy and security problems and are based on high overhead asymmetric encryption algorithms. This paper proposes a privacy-preserving keyword search scheme for searching over encrypted data. To avoids the high computational cost of asymmetric encryption, the proposed scheme employs symmetric encryption and Bloom filter. Experimental results demonstrate that the proposed searchable encryption algorithm is secure and lightweight, and it has the ability to perform a keyword search over encrypted data without decrypting them.

Keywords: cloud computing, privacy, encryption, Bloom filter, confidentiality

1.Introduction

Cloud computing is considered as the next generation paradigm in computation. In the cloud computing environment, data is located in different places. Data security and privacy protection are the two main factors of the user's concerns about cloud technology. Also, data security and privacy protection are becoming more important for the future development of cloud computing technology in government, industry, and business.

Received: Jan 19, 2020 Revised: April 28, 2020 Accepted: June 15, 2020

DOI: 10.5281/zenodo.3906950

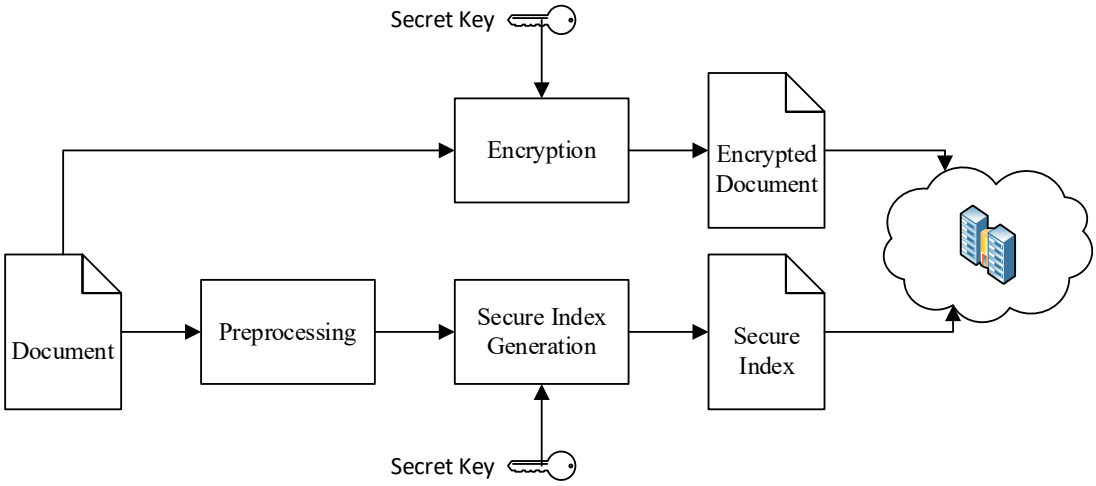

Figure 1. General architecture of the proposed scheme

The explanation of "cloud computing" from the National Institute of Standards and Technology (NIST) [2] is that cloud computing enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. According to the explanation, cloud computing provides convenient on-demand network access to a shared pool of configurable computing resources. Resources refer to computing applications, network resources, platforms, software services, virtual servers, and computing infrastructure.

Currently, most data owners outsource their data to cloud storage services, so that they can get access to these data as they want. It is clear that cloud computing takes the economic savings and management flexibility for data owners. However, most data owners do not want to store their sensitive data such as financial records into cloud storage on account of data privacy. The cloud storage service is semi-trusted, which may leak data privacy. So, it is necessary for data owners to encrypt their data before outsourcing it. However, encryption may reduce the efficiency of data utilization, especially for some services that are based on plaintext keyword searches. Hence, searchable encryption allows data owners to get access to relevant data by searching their encrypted data.

1.1. Research methodology

The proposed searchable encryption scheme involves two entities, which are the data owner and the cloud storage service. The data owner is the entity such as people, companies, and organizations, who want to outsource a document set DS into the cloud. But before that, the data owner must encrypt DS to ensure the confidentiality and the privacy of the data. Also, the related secure search index should also be built to achieve the search capability on the encrypted data. Then the data owner will outsource the encrypted document set EDS and the corresponding search index to the remote cloud server. When the data owner wants to search the relevant documents from the cloud server with the query keywords, the data owner must generate the corresponding request trapdoor Q through the proposed Bloom filter-based search method. After the cloud server receives the request trapdoor Q, it will check the corresponding Bloom filters associate with all documents in DS. Then the cloud server will return the top-k most relevant encrypted documents to the data owner in terms of word frequency and bag of words (BoW) model [1], see Figure 1.

1.2. Organization of the paper

The remainder of the paper is organized as follows. Section 2 discusses related works. In Section 3, the preliminaries for the proposed scheme. In Section 4, the proposed privacy-preserving keyword search scheme is described. The discussion and details of experimental results and analysis are presented in Section 5. Finally, Section 6 concludes the paper and discusses future work.

DOI: 10.5281/zenodo.3906950

2. Related work

To support the secure keyword search over the outsourced encrypted cloud data, researchers have proposed many Searchable Encryption (SE) schemes:

The first searchable encryption scheme was proposed by Song et al. [2]. Their scheme utilizes two-layered encryption, which can guarantee the correctness of the trapdoor. Although this scheme is proven to be secure, it is based on a weak security model. To address the drawbacks of Song et al [2] some novel searchable encryption schemes [3-10] were proposed. These schemes build encrypted indices for searching. Boneh et al [5] presented the first PKC-based SE mechanism, where anyone with the public key can write to the data stored on the server, but only authorized users with a private key could search. Later, some schemes [11-13] achieved a single keyword rank search using order-preserving techniques.

Fu et al. [14] proposed a synonym expansion of document keywords and realized the synonym-based multi-keyword ranked search scheme. Xia et al. [15] proposed a multi-keyword semantic ranked search scheme where the inverted index for documents and the semantic relationship library for keywords are adopted. Fu et al. [16] proposed a different semantic-aware ranked search scheme that adopts the concept hierarchy and the semantic relationship between concepts.

Cao et al. [17] proposed a privacy-preserving multi-keyword ranked search scheme, in which documents and queries were represented as vectors of dictionary size. Since the importance of the different keywords wasn't considered, this scheme was not accurate enough. Moreover, the search complexity was linear with the size of the data collection. To improve search efficiency and rank accuracy, Sun et al.[18], [19] constructed a searchable tree-based index structure. Their scheme achieved better-than-linear search efficiency. Zhang et al. [20] proposed a multi-keyword ranked search scheme in a multi-owner model, in which an "Additive Order Preserving Function" was adopted to retrieve the most relevant search results.

3. Preliminaries

This section briefly introduces some preliminary material and notations used in this paper. Here, we consider various text similarity measures that may be used in the proposed watermarking method for watermark verification. This section also introduces the concept of a Bloom filter, which is employed to reduce the size of the generated watermark.

3.1 Text preprocessing

Much of the text preprocessing methods have their roots in natural language processing. Text preprocessing takes an input of raw text and returns cleansed tokens. Tokens are single words or groups of words that are tallied by their frequency. The preprocessing process includes:

1. Unitization and tokenization.
2. Standardization and cleansing or text data cleansing.
3. Stop word removal.
4. Stemming or lemmatization.

The stages along the pipeline standardize the data, thereby reducing the number of dimensions in the dataset. There is a balance between retained information and reduced complexity in the choices made during the process. Each step removes unnecessary information from the original text.

3.1.1 Unitization and tokenization

The first step involves the choice of the unit of text to analyze and the separation of the text based on the unit of analysis. This unit could be a word. Tokenization is done under the assumption of the bag-of-words (BoW)) model

DOI: 10.5281/zenodo.3906950

[1], meaning that the grammar and order of the text in a document are not considered in building the quantitative representation of the qualitative text data.

The bag-of-words model is a simplifying representation used in natural language processing. In this model, a text (a document) is represented as the bag of its words, After transforming the text into a "bag of words" we can characterize the text by the term frequency, that is the number of times a term appears in the text.

3.1.2 Standardization and cleansing

Standardize and cleaning the tokens is done by making the terms in each of the documents different from each other. Our first step is to convert the terms in the text to lower case. Following this conversion, we want to remove numbers, punctuation e.g. periods and exclamation points, and special characters.

3.1.3 Stop word removal

This process drops filler words or stop words. According to the Oxford English Dictionary, and the, be, to, and of are the most common words in the English language. a, and, has, he, is, my, the, and was were removed from the documents.

3.1.4 Stemming

Stemming involves breaking words down to their root word and involves the removal of a word's suffix to reduce the size of the vocabulary. This method combines words that contain the same root into a single token to reduce the number of unique tokens. As an example, let's use the root train. Train has several forms, including:

— Train

— Trains

— Trained

— Training

— Trainer

3.2 Bloom filters

The proposed scheme method employs a Bloom filter (BF) [21] to improve the keyword search efficiency. A BF is a space- and time-efficient probabilistic data structure introduced by Burton Bloom in 1970. It is an array of m bits (where all bits are initialized to 0) constructed using k hash functions ( ) with values in the range of [22]. To insert a string into a BF, we compute the hash functions of the string and set the corresponding bits in the BF bit string to 1. To examine the membership of string s, the hash functions of s are computed, and it is verified that all bits at positions are set to 1. If any bit returns zero, then s is not a member of the BF. A membership query will return a false positive result if all corresponding bits are equal to 1 even though the element of interest was not inserted into the BF. On the other hand, a BF membership query will never return a false negative. The probability of a false positive depends on (i) the number of bits m used in the BF, (ii) the number of hash functions k, and (iii) the number of elements n stored in the BF. It has been proven that the probability of a false positive can be computed as follows [23]:

DOI: 10.5281/zenodo.3906950

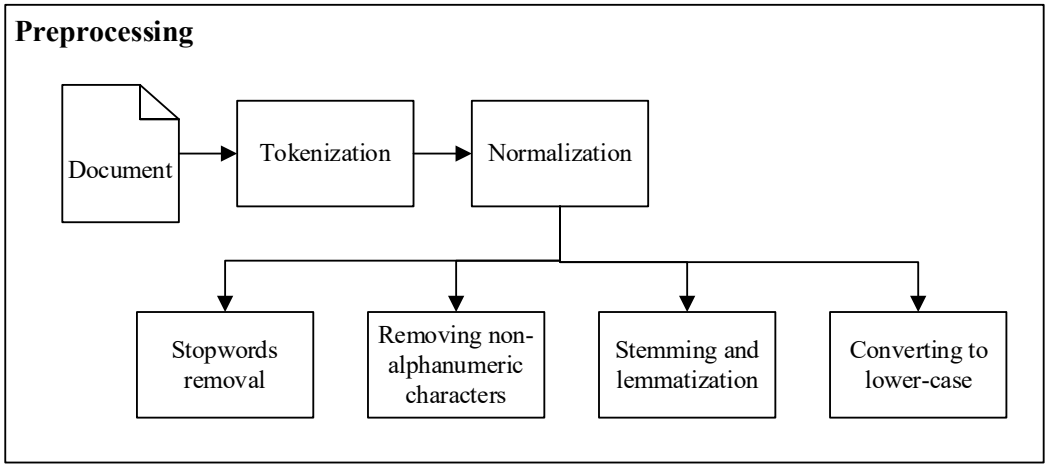

Figure 2. Document preprocessing

$$fp = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \approx \left(1 - e^{-kn/m}\right)^{k} \quad (1)$$

The optimal number of hash functions $\bar{k}$ that minimizes fp can be computed by taking the derivative of the above equation:

$$\bar{k} = \frac{m}{n}.\ln(2) \quad (2)$$

The required number of bits m can be computed by taking the natural logarithm of both sides of Eq. (1):

$$m = -n \ln(fp) / \ln(2)^2 \quad (3)$$

4. Proposed scheme

This section presents the proposed privacy-preserving keyword search scheme for searching over encrypted data. In order to overcome the high computation cost of traditional solutions like homomorphic encryption schemes [24-27] , and to achieve liner search time, our scheme employs Bloom filter to build the search indexes of the outsourced encrypted documents. The proposed scheme uses symmetric encryption, which provides low computational cost over schemes based on the public key and homomorphic encryption. The scheme provides a simple encrypted document ranking capability using keyword frequency. Using the Bloom filter, the proposed scheme measure how often a keyword appears on a given encrypted document. The more frequently a keyword appears in the encrypted document, the higher the document is ranked.

We assume that the data owner wants to outsource her document set to the CSP. First, the data owner must encrypt the document set using a symmetric encryption algorithm e.g. AES, and symmetric key K known only to the data owner for the consideration of confidentiality. Next, the Bloom filter based secure index (BFSI) is built based on the extracted BoW from the original document set. The following operations are performed to achieve the design goals of the proposed scheme:

4.1 Document preprocessing

One of the most important steps of the proposed scheme is document preprocessing. It transforms text into a more digestible form so that the following phases can perform better. It includes the following processes and each process removes unnecessary information from the original text document: tokenization, normalization, stemming and lemmatization, See Figure 2.

DOI: 10.5281/zenodo.3906950

Algorithm 1: Tokenization

Input: *Document*
Output: *List of tokens*.

```
Split Document into Paragraphs;
foreach Paragraph in Document do
    Split Paragraph into Sentences
    foreach Sentence in Paragraph do
        Split Sentence into tokens;
    end foreach
end foreach
return Tokens
```

Algorithm 2: Stop Words Removal

Input: *Document*
Output: *List of tokens*.

```
foreach token in List of tokens (T) do
    if token in T then
        Remove token from T;
    end if
end foreach
return Updated T
```

4.1.1 Tokenization

The tokenization process is the first process of the preprocessing phase. This process splits text into its basic units. This unit may be paragraphs, sentences, words, characters, numbers or any other appropriate unit [28]. The main basic units in tokenization are paragraphs, sentences, and words [29]. Algorithm 1 shows the process of tokenization.

4.1.2 Normalization

Normalization: is the process of making a text more consistent. Text normalization includes the following procedures:

- Removal of stop words (e.g., “a”, “the”, “is”, and “are”). These insignificant words appear in an article or web page, which is non-informative. In addition, they shorten the document’s length, thus affecting the weighing process. Moreover, they do not help to identify document topics and reducing features [30], see Algorithm 2.
- Removal of punctuation, numbers, white spaces, and single-character and double-character words.
- Conversion of all letters to lower case. In this step, any capital letters are converted to lower case. Without this conversion, two identical words would erroneously be considered two different terms.
- Spelling correction using tools that automatically correct misspellings by utilizing a detailed database of the language.

4.1.3 Stemming and lemmatization

The final step is to perform either stemming or lemmatization on the documents. Stemming and lemmatization involve breaking words down to their root word. Stemming involves the removal of a word’s suffix to reduce the size of the vocabulary [31]. Lemmatization is similar to stemming, except it incorporates information about the term’s part of speech [32]. Both methods combine words that contain the same root into a single token to reduce the number of unique tokens within the analysis set. Words with a common root often share a similar meaning. These words are then grouped into one token.

4.2 Construction of Bloom filter based secure index

Suppose that a document *D* and a secret key *K* are known to the data owner. The data owner follows the following step to construct the document’s secure index SI, see Figure 3:

Step 1. The given document D is first preprocessed as described in Section **Error! Reference source not found.**. The output document is denoted by $\overline{D_1}$.

Step 2. The BoW mode is applied to $\overline{D_1}$ to generate $W = \{w_1, w_2, \ldots, w_n\}$, where w_k is a unique word. Also, the word frequency is computed: $F = \{f_1, f_2, \ldots, f_n\}$ where f_i is the frequency at which w_i appears in the document. Finally, $f_{\max} = \max\{F\}$ is computed.

Step 3. A BF of size *n* and false-positive probability *fp* = *p* is created.

DOI: 10.5281/zenodo.3906950

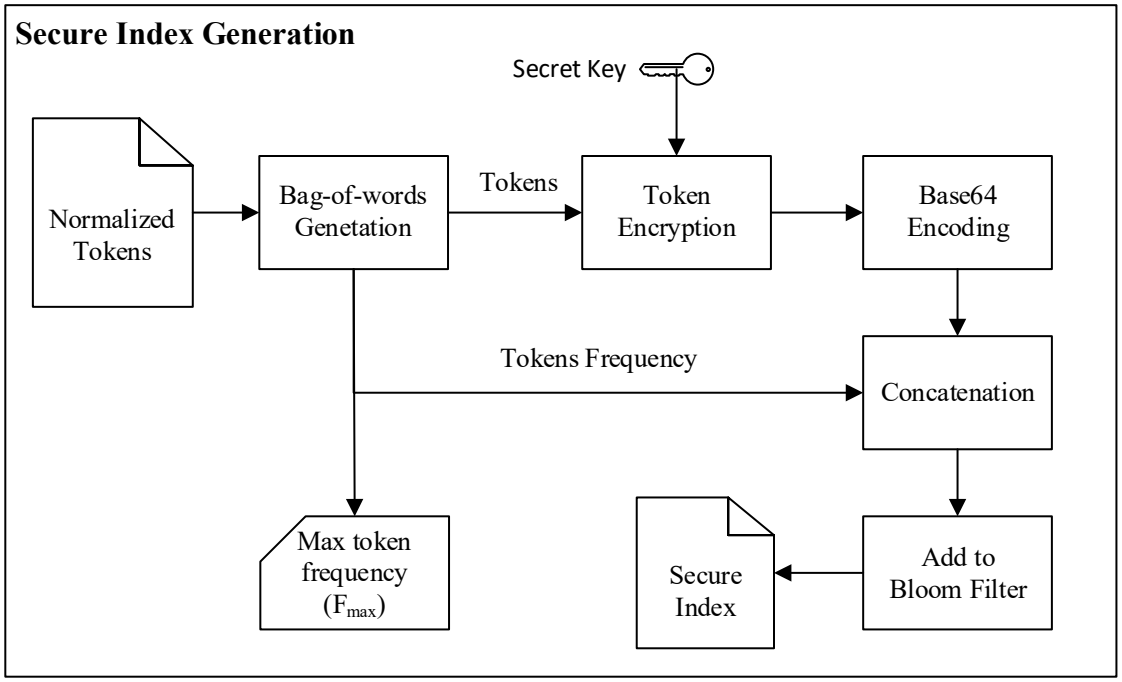

Figure 3. Secure index generation

Step 4. Each pair (w_i, f_i) is added into the BF by first encrypting w_i using the secret key K i.e. $cw_i = E_K(w_i)$, where E is symmetric encryption algorithm e.g. AES. Next, cw_i is encoded to base64 code and concatenated with f_i into a single string $sw_i = (base64(cw_i) + f_i)$, Finally, the final string sw_i is inserted into the BF.

Step 5. The algorithm returns the frequency of the most repeated word ($f_{\max}$) and BF.

Step 6. The data owner outsources $f_{\max}$, secure index (BF), and $CD = E_K(D)$ to the cloud storage service.

4.3 Privacy-preserving keyword search

The data owner can perform secure keywords search on the encrypted document efficiently by following the following steps, see Figure 4:

Step 1. For a given keyword set *W*, the data owner applies the preprocessing phase to each word in *W*. The output keyword set is denoted by $\overline{W}$.

Step 2. Each pair $\overline{w_i} \in \overline{W}$ is encrypted using the secret key *K*. Next, each encrypted $\overline{w_i}$ is encoded to base64 code ($\overline{sw_i}$) and sent to the cloud storage service.

Step 3. Each encode word $\overline{sw_i}$ is first converted into a string in the form $(\overline{sw_i}\_g)$. Next, the BF of each encrypted document in the document et DS is queried for the presence or absence of word frequency g using all possible word frequencies, i.e., $\left\{\overline{sw_i}\_g\right\}_{g=1}^{f_{\max}}$. If the query returns true for one of the word frequencies (e.g., $(\overline{sw_i}\_g), 1 \le g < f_{\max}$), then the corresponding keyword is present in the document with frequency *g*. The more frequently a keyword appears in the encrypted document, the higher the document is ranked.

DOI: 10.5281/zenodo.3906950

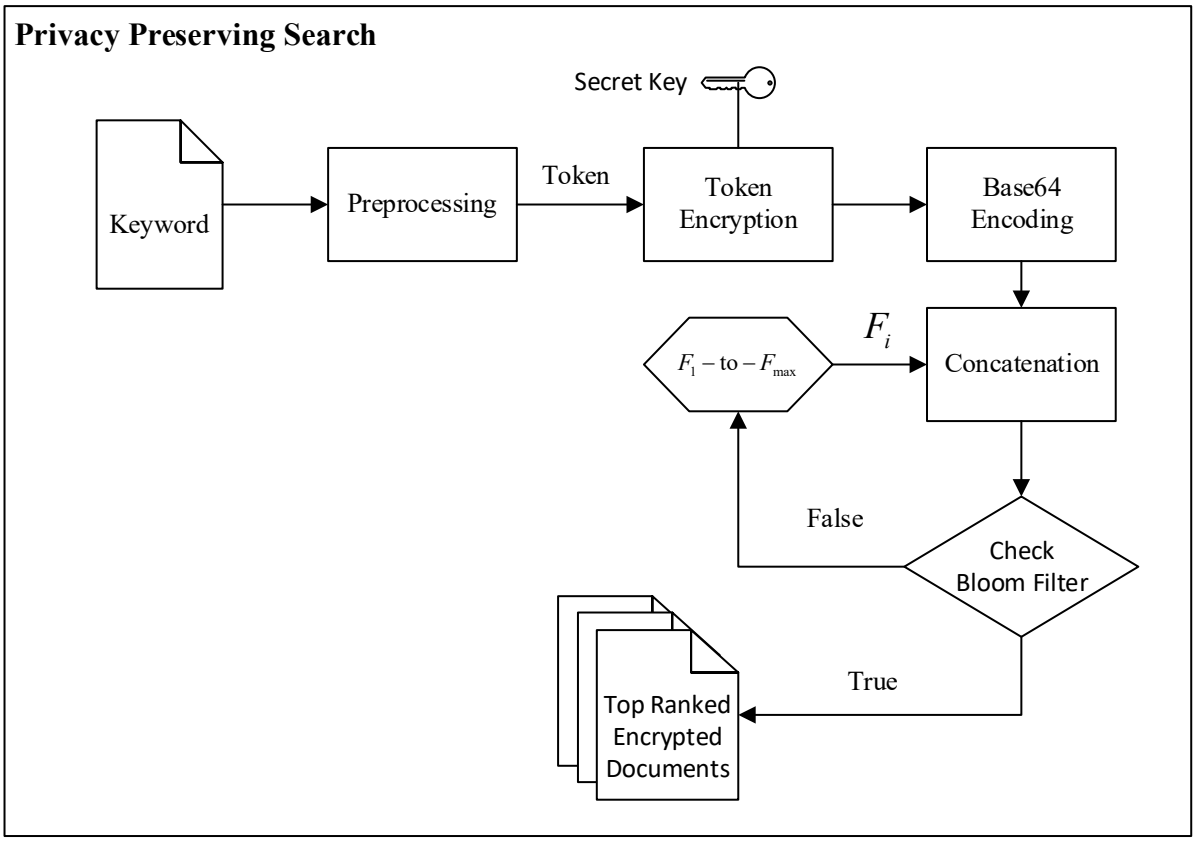

Figure 4. Privacy preserving keyword search

5. Results and discussion

To evaluate the proposed scheme, a prototype implementation was created as a python application running on a PC with the Windows 10 operating system, a Core i7 @2.60 GHz CPU, and 16 GB of RAM. The US Inaugural Speeches dataset [33] was used in the evaluation process. This dataset corpus actually contains dozens of text files — one per address. Symmetric encryption was performed using the AES algorithm with 128 bits key. Also, a Bloom filter with a false-positive probability of 0.1 is used. The proposed scheme was evaluated using the following metrics for both the data owner and the cloud storage service: computation cost, storage overhead, and communication cost.

5.1 Computation cost

The computation cost represents the processing delays of the document preprocessing, cryptographic and encoding operations, and Bloom filter operations at both the data owner and the cloud storage service. To evaluate the computation cost, the proposed prototype is applied on each file of the 58 files in the dataset with respect to the preprocessing, secure index construction and secure search operation. Table 1 and Figure 5 show the result of the evaluation. As shown in Figure 5, the index construction time is almost constant with respect to file size this is because the number of words in the BoW model tends to be constant as the file size increases. The secure search operation was less than 1 millisecond on both the data owner side and the cloud storage service side

DOI: 10.5281/zenodo.3906950

Table 1. Computation cost of the proposed scheme

File number	File Size (bytes)	Preprocessing (ms)	Secure Index Construction (ms)	File number	File Size (bytes)	Preprocessing (ms)	Secure Index Construction (ms)
1	791	45	4	30	12018	1354	47
2	3039	236	13	31	12250	1123	38
3	3926	256	162	32	12349	756	43
4	5568	399	24	33	12523	845	42
5	6503	547	23	34	12908	729	40
6	6605	374	27	35	13408	823	68
7	6817	434	54	36	13439	1172	49
8	6873	417	25	37	13679	828	35
9	7001	388	24	38	13735	843	43
10	7058	477	22	39	13877	783	50
11	7157	397	27	40	13955	1496	150
12	7571	478	25	41	14561	931	45
13	7618	699	33	42	14938	861	45
14	7734	496	27	43	16815	1040	52
15	8193	516	37	44	17741	1333	50
16	8395	617	43	45	17767	1112	55
17	8449	591	28	46	19887	1150	47
18	8619	476	38	47	20081	1411	65
19	9053	724	67	48	20298	1994	68
20	9114	679	35	49	21017	1340	54
21	9190	1109	80	50	21764	1282	49
22	9563	679	32	51	23417	1287	69
23	9991	666	23	52	23659	1480	59
24	10137	580	35	53	23949	1475	58
25	10145	587	42	54	26179	1546	76
26	10607	637	34	55	26326	1951	93
27	10903	929	45	56	28716	1995	66
28	11624	727	37	57	32164	1963	69
29	11949	801	45	58	49700	2856	95

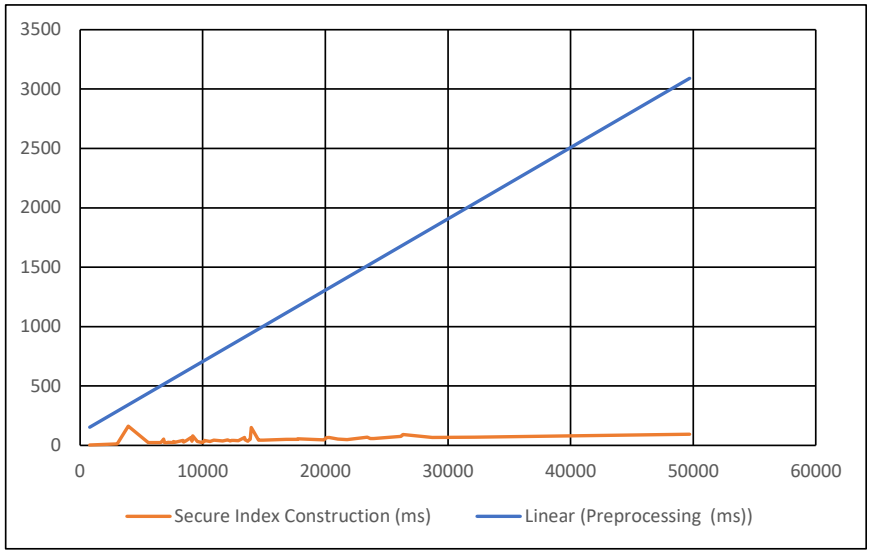

Figure 5. Preprocessing and secure index construction time

DOI: 10.5281/zenodo.3906950

Table 2. Storage cost of the proposed scheme

File Number	File Size (bytes)	Secure Index Size (bytes)	Stoarge Overload (%)	File Number	File Size (bytes)	Secure Index Size (bytes)	Stoarge Overload (%)
1	791	34	4	30	12018	324	3
2	3039	105	3	31	12250	328	3
3	3926	146	4	32	12349	370	3
4	5568	171	3	33	12523	326	3
5	6503	197	3	34	12908	349	3
6	6605	222	3	35	13408	367	3
7	6817	230	3	36	13439	408	3
8	6873	224	3	37	13679	331	2
9	7001	238	3	38	13735	380	3
10	7058	212	3	39	13877	357	3
11	7157	242	3	40	13955	394	3
12	7571	222	3	41	14561	379	3
13	7618	236	3	42	14938	353	2
14	7734	221	3	43	16815	409	2
15	8193	230	3	44	17741	439	2
16	8395	227	3	45	17767	443	2
17	8449	243	3	46	19887	442	2
18	8619	272	3	47	20081	509	3
19	9053	264	3	48	20298	509	3
20	9114	267	3	49	21017	443	2
21	9190	267	3	50	21764	448	2
22	9563	281	3	51	23417	565	2
23	9991	207	2	52	23659	529	2
24	10137	316	3	53	23949	512	2
25	10145	295	3	54	26179	587	2
26	10607	303	3	55	26326	529	2
27	10903	327	3	56	28716	555	2
28	11624	318	3	57	32164	591	2
29	11949	363	3	58	49700	763	2

5.2 Storage overhead

In the proposed scheme the data owner outsources the encrypted file, the secure index, and the frequency of the most repeated word ($f_{\max}$) to the cloud storage service. Hence there is not any storage overhead on the data owner side. However, the cloud storage service store the Bloom filter based secure index beside the encrypted file. Table 2 and Figure 6 show the storage overhead on the cloud service storage due to the secure index. As shown in Figure 6, the storage overhead decreases as the document size increases this is because the number of words in the BoW model tends to be constant as the file size increases. As shown in Table 2, the storage overhead on the cloud storage service side approaches 2% as the document size increases.

5.3 Communication cost

The communication cost in our proposed scheme, during the privacy-preserving search, consists of two parts: the search query, sent from the data owner to the cloud storage service which is the encrypted keywords encoded into base64, and the response sent back from the cloud storage service to the data owner which is the top k-ranked

DOI: 10.5281/zenodo.3906950

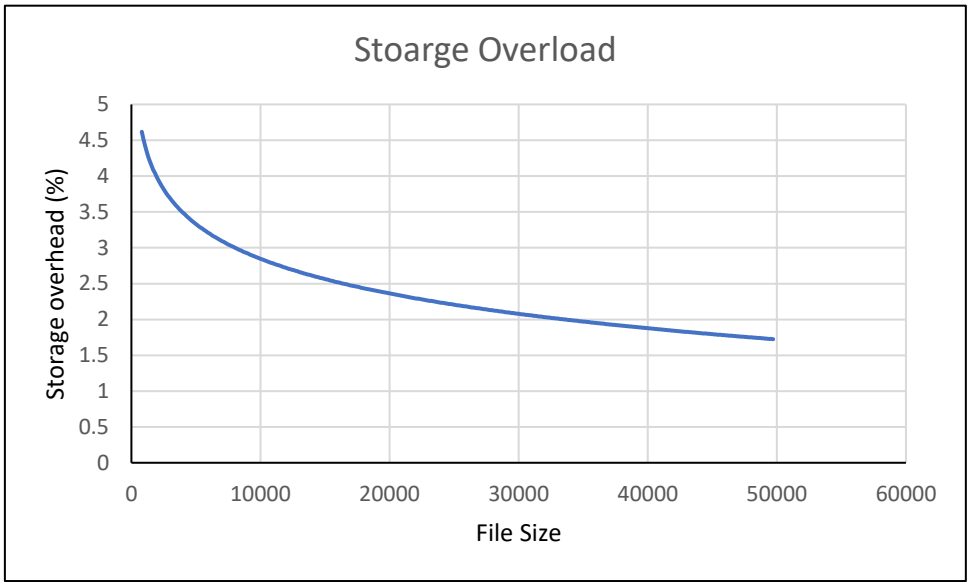

Figure 6. Storage overhead at the cloud storage side

encrypted documents. It is clear that the extra communication cost will be on the data owner side due to base64 encoding which adds about 33% of the original keywords [34]. However, this increase does not affect the storage space on both sides of the communication.

6. Conclusion and future work

One of the main challenging tasks in cloud computing is to ensure the efficiency of the search under the premise of preserving the privacy of the search query and the search results. Most of the existing keyword ranked search over encrypted data is based on the high overhead public key and homomorphic encryption. In this paper, we propose a privacy-preserving keyword ranked search over encrypted data. The proposed scheme avoids the high computational cost of asymmetric encryption by employing symmetric encryption and Bloom filter to encrypt the file and to generate the secure index of the document, respectively. The proposed scheme provides keyword-query privacy by encrypting the keyword using a secret key known only to the data owner followed by base64 encoding. This prevents the cloud storage service from disclosing the keywords. The cloud storage service uses the encrypted keyword to query the Bloom filter associated with each encrypted file to get the top k-ranked encrypted documents that satisfy the encrypted query. This provides privacy and confidentiality for both the query and the document. The experimental results show that the proposed scheme is efficient in terms of computation, storage, and communication costs at both the data owner side and the cloud storage service. As future work, we are interested in improving the proposed Bloom filter-based privacy-preserving cloud search to support supports dynamic update operation e.g. insertion and deletion operation of the document text.

References

1. M. J. N. L. E. Sanderson, "Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. ISBN-13 978-0-521-86571-5, xxi+ 482 pages," vol. 16, no. 1, pp. 100-103, 2010.
2. D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, 2000, pp. 44-55: IEEE.
3. Z. Xia, X. Wang, X. Sun, Q. J. I. t. o. p. Wang, and d. systems, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," vol. 27, no. 2, pp. 340-352, 2015.
4. R. Curtmola, J. Garay, S. Kamara, and R. J. J. o. C. S. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," vol. 19, no. 5, pp. 895-934, 2011.

DOI: 10.5281/zenodo.3906950

5. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in International conference on the theory and applications of cryptographic techniques, 2004, pp. 506-522: Springer.
6. Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. J. I. T. o. C. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," vol. 98, no. 1, pp. 190-200, 2015.
7. P. Van Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker, "Computationally efficient searchable symmetric encryption," in Workshop on Secure Data Management, 2010, pp. 87-100: Springer.
8. Z. Fu, K. Ren, J. Shu, X. Sun, F. J. I. t. o. p. Huang, and d. systems, "Enabling personalized search over encrypted outsourced data with efficiency improvement," vol. 27, no. 9, pp. 2546-2559, 2015.
9. S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in Proceedings of the 2012 ACM conference on Computer and communications security, 2012, pp. 965-976.
10. J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in 2010 Proceedings IEEE INFOCOM, 2010, pp. 1-5: IEEE.
11. S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+ r: Top-k retrieval from a confidential index," in Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, 2009, pp. 439-449.
12. C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in 2010 IEEE 30th international conference on distributed computing systems, 2010, pp. 253-262: IEEE.
13. C. Wang, N. Cao, K. Ren, W. J. I. T. o. p. Lou, and d. systems, "Enabling secure and efficient ranked keyword search over outsourced cloud data," vol. 23, no. 8, pp. 1467-1479, 2011.
14. Z. Fu, X. Sun, N. Linge, and L. J. I. T. o. C. E. Zhou, "Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query," vol. 60, no. 1, pp. 164-172, 2014.
15. Z. Xia, Y. Zhu, X. Sun, and L. J. J. o. C. C. Chen, "Secure semantic expansion based search over encrypted cloud data supporting similarity ranking," vol. 3, no. 1, pp. 1-11, 2014.
16. Z. Fu, L. Xia, X. Sun, A. X. Liu, G. J. I. T. o. I. F. Xie, and Security, "Semantic-aware searching over encrypted data for cloud computing," vol. 13, no. 9, pp. 2359-2371, 2018.
17. N. Cao, C. Wang, M. Li, K. Ren, W. J. I. T. o. p. Lou, and d. systems, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," vol. 25, no. 1, pp. 222-233, 2013.
18. W. Sun et al., "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, 2013, pp. 71-82.
19. W. Sun, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving keyword search over encrypted data in cloud computing," in Secure cloud computing: Springer, 2014, pp. 189-212.
20. W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. J. I. T. o. C. Zhou, "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," vol. 65, no. 5, pp. 1566-1577, 2015.
21. M. Klonowski and A. M. Piotrowska, "Light-weight and secure aggregation protocols based on Bloom filters," Computers & Security, vol. 72, pp. 107-121, 2018/01/01/ 2018.
22. A. Kumar and A. R. J. W. N. Pais, "A new hybrid key pre-distribution scheme for wireless sensor networks," Wireless Networks, vol. 25, no. 3, pp. 1185-1199, 2019.
23. S. Geravand and M. Ahmadi, "Bloom filter applications in network security: A state-of-the-art survey," Computer Networks, vol. 57, no. 18, pp. 4047-4064, 2013/12/24/ 2013.
24. M. Kim, H. T. Lee, S. Ling, B. H. M. Tan, and H. Wang, "Private Compound Wildcard Queries Using Fully Homomorphic Encryption," IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 5, pp. 743-756, 2019.
25. D. Reis, M. T. Niemier, and X. S. Hu, "A Computing-in-Memory Engine for Searching on Homomorphically Encrypted Data," IEEE Journal on Exploratory Solid-State Computational Devices and Circuits, vol. 5, no. 2, pp. 123-131, 2019.

DOI: 10.5281/zenodo.3906950

26. J. Ji and M. Shieh, "Efficient Comparison and Swap on Fully Homomorphic Encrypted Data," in 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, pp. 1-4.
27. J. Liu, J. Han, and Z. Wang, "Searchable Encryption Scheme on the Cloud via Fully Homomorphic Encryption," in 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), 2016, pp. 108-111.
28. F. Alotaiby, I. Alkharashi, and S. Foda, "Processing large Arabic text corpora: Preliminary analysis and results," in Proceedings of the second international conference on Arabic language resources and tools, 2009, pp. 78-82: Citeseer.
29. V. Cavalli-Sforza and I. Zitouni, "Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources," in Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources, 2007.
30. F. Amato, G. Cozzolino, M. Giacalone, F. Moscato, F. Romeo, and F. Xhafa, "A Hybrid Approach for Document Analysis in Digital Forensic Domain," Cham, 2019, pp. 170-179: Springer International Publishing.
31. M. F. J. P. Porter, "An algorithm for suffix stripping," vol. 14, no. 3, pp. 130-137, 1980.
32. V. J. A. D. Yatsko and M. Linguistics, "Methods and algorithms for automatic text analysis," vol. 45, no. 5, pp. 224-231, 2011.
33. Adhokshaja, "US Inaugural Speeches," Adhokshaja, Ed., ed. https://data.world/adhokshaja/us-inaugural-speeches, 2017.
34. S. Josefsson, "RFC3548: The Base16, Base32, and Base64 Data Encodings," ed: RFC Editor, 2003.

DOI: 10.5281/zenodo.3906950