Efficient multi-keyword similarity search over encrypted cloud documents

Ayad I. Abdulsada, Dhafer G. Honi, Salah Al-Darraji
Department of Computer Science, College of Education for Pure Science, University of Basrah, Iraq

Article Info

Article history:

Received Dec 6, 2020
Revised Apr 12, 2021
Accepted Jun 16, 2021

Keywords:

Cloud computing
Multi-keywords ranking search
Privacy preserving
Searchable encryption
Simhash

ABSTRACT

Many organizations and individuals are attracted to outsource their data into remote cloud service providers. To ensure privacy, sensitive data should be encrypted before being hosted. However, encryption disables the direct application of the essential data management operations like searching and indexing. Searchable encryption is a cryptographic tool that gives users the ability to search the encrypted data while being encrypted. However, the existing schemes either serve a single exact search that loss the ability to handle the misspelled keywords or multi-keyword search that generate very long trapdoors. In this paper, we address the problem of designing a practical multi-keyword similarity scheme that provides short trapdoors and returns the correct results according to their similarity scores. To do so, each document is translated into a compressed trapdoor. Trapdoors are generated using key based hash functions to ensure their privacy. Only authorized users can issue valid trapdoors. Similarity scores of two textual documents are evaluated by computing the Hamming distance between their corresponding trapdoors. A robust security definition is provided together with its proof. Our experimental results illustrate that the proposed scheme improves the search efficiency compared to the existing schemes. Furthermore, it shows a high level of performance.

Corresponding Author:

Ayad I. Abdulsada
Department of Computer Science, College of Education for Pure Science
University of Basrah, Iraq
Email: ayad.abdulsada@uobasrah.edu.iq

1. INTRODUCTION

Cloud computing is a promising technology that supports cost-effective solutions for storing and processing large datasets. For this reason, individuals and organizations with constrained-resource machines tend to outsource their data collections to such professional power servers. However, such outsourced service may raise main concerns towards users privacy, where personal data should be preserved [1]. Such data may include E-mail, medical information, private videos, and photos. Therefore, users employ encryption to protect the privacy of their confidential data. Unfortunately, encryption disables traditional keyword search operations on remote data. Searchable encryption schemes allow preforming search over encrypted data at the server side without decryption. Like search over plaintext data, searchable encryption methods build a searchable index from the entire dataset, such that during the search, only trapdoors generated using a secret keys can match index entries to get relevant results. Index contents should reveal nothing to the adversary server. Index-based search not only enhances search efficiency, but also isolates data and index encryption schemes. Under the setting of searchable symmetric encryption (SSE) schemes, the encrypted data are uploaded and retrieved by the same party. Most of SSE methods focus on serving exact single keyword search queries [2–6]. Some methods [7, 8] focus on single keyword fuzzy search. The results of such schemes are too large especially when huge data collections are used. Users can filter the results locally by sending a list of single-keyword queries and get the intersection of the returned results. This scheme is inefficient and requires intensive computations for the user. Another solution is to find the intersection at the server side and ask the latter to return the final results to the user. This enables the server from learning undesirable information such as the intersection patterns for all keywords. Therefore, it is necessary to have SSE schemes that enable the submission of search queries with a conjunction of several keywords. Such scheme reduces the computation burden of users since only the best matching documents are retrieved. In this paper, we propose a new privacy preserving yet efficient multi-keyword similarity search scheme that returns the relevant documents according to their similarity scores even in case of providing misspelled keywords.

Contributions: Contributions are briefly described as follows: Firstly, we employ Simhash function to design privacy-preserving scheme that supports multi-keyword similarity search. Such a function transforms the textual documents into a fixed size vectors, such that two similar documents lead to two similar vectors. Secondly, we introduce adaptive semantic security definition and prove that the proposed scheme satisfies this definition. Thirdly, we utilize a ranking method that employs XOR operation between two bit vectors. Finally, we implement the proposed scheme and show its efficiency and effectiveness.

Organization: The paper is outlined as follows. Section 2 summarizes the previous works. Section 3 provides the problem description and illustrates its security definition. Section 4 shows the basic steps of the proposed scheme. Section 5 reports the formal prove of the scheme. Section 6 shows the results of the conducted experiments, and section 7 concludes our work.

2. RELATED WORK

Single keyword SSE schemes: The problem of searching over encrypted data is addressed by various works in the literature. The majority of these works focus on handling a single keyword search requests. Under such a setting, searchable index is generated and encrypted before being uploaded. Search process scans the encrypted index to identify the documents that includes the provided queries. Goh *et al.* [2] introduced a Bloom filter based construction, where the unique keywords of a given document are hashed into a single filter. The major problem of this work is that it may return false positive results. Chang *et al.* [5] provided simulator-based security definition, which is stronger than [2]. Their index construction searches without false positive results. Later on, Curtmola *et al.* [6] employed the inverted index structure to describe the entire collection. Here, search time is logarithmic in the total number of the search keywords stored on the server. Strizhov and Ray [9] and Cash *et al.* [3] developed new schemes that give the data owner the ability to update his collection with a minimum leak. Dynamic SSE constructions with advanced security definitions are presented in [10–15]. The main disadvantage of the above mentioned schemes is that they use only single keyword search which return massive number of results, where most of them are not relevant to the users.

Single Similarity SSE schemes: Some SSE schemes enhance query richness to support similarity search, where the scheme can retrieve the correct results even with providing misspelled keywords. In the work of [7], each keyword is expanded by listing all its variants that are within a specific edit distance. All keyword variants are stored in the searchable index, which require more computation and communication costs. Kuzu *et al.* [8] employ the Minhash technique to capture the Jaccard distance between two sets. Such schemes support only single keyword similarity search.

Multy-keyword SSE schemes: To enhance search functionality, multi-keyword search is used to retrieve only the relevant data items. Cao *et al.* [16] proposed the first scheme that performs ranked multi-keyword search, where each document is represented as a binary vector. The size of this vector is determined by the total number of keywords in the vocabulary. Vectors are protected by multiplication with randomly generated matrices. This scheme requires from the data users to know the position of each keyword to generate valid search requests. Furthermore, binary vectors ignore the importance of each keyword within the provided document. Additionally, similarity scores are defined as the number of matched keywords between two vectors, which is not standard operation for ranking the returned results. Cash *et al.* [4] designed an efficient construction that supports conjunctive queries. Such a construction returns only the documents that match all items of the provided query. However, pairing-based solutions require intensive computational costs. Örencik *et al.* [17] proposed an efficient ranked multi-keyword search scheme. In this scheme, a fixed-size vector is generated for each document. The underline method does not allow for ranking results. The work of [18] described an efficient scheme that solved the problem of document ranking with multi-keyword search. They employed *Minhash* method for generating fingerprint items for each file. The items of the entire collection are used to construct an inverted index. However, such scheme assumes the existence of two non-colluding servers to rank the returned results. New security improvements are presented in [19]. Our scheme uses only one server which performs all ranking tasks. Strizhov *et al.* [9] used inner product similarity and *tf-idf* based ranking model. Such a model uses only one server to answer multi-keyword queries, where results are returned without sorting. Such a model uses inefficient fully homomorphic encryption scheme. Baldimtsi *et al.* [20] designed recently a tool for sorting an encrypted data. Such a tool was successfully utilized to solve the problem of ranked search over encrypted data, where two servers are needed. Recently, a secure multi-keyword ranked search that release dynamic search functionality is proposed in [21], [22]. Wang *et al.* [23] proposed a secure scheme that protects the search pattern. Recently, Rani *et al.* [24] designed a tree-based index to solve the problem of secure search.

3. PROBLEM DESCRIPTION AND SECURITY DEFINITION

In this section, the notations, formal description of the problem, and the security definition are presented.

3.1. Notations

Let λ be the security parameter. $D = \{D_1, D_2, \ldots, D_n\}$ is a document collection of size n, $D_i = \{w_1, ..., w_m\}$ is a set of m distinct keywords, $C = \{C_1, C_2, \ldots, C_n\}$ is the encrypted collection, $id(C_i)$ is the identifier of document C_i, $|C_i|$ is the size of the encrypted document C_i, $Q = \{Q_1, Q_2, \ldots, Q_q\}$ is a collection of q successive queries, where $Q_i = \{w_1, ...w_s\}$ is a list of s keywords, $DB(kw)$ is the collection of documents that match the multi-keyword query kw, and $SI = \{SI_1, ..., SI_n\}$ is the index, where SI_i is the document index of D_i. KD and KS represent secret keys for encrypting document collection and searchable index, respectively. The occurrence of keyword w_i within a given document is denoted by tf_i, whereas T is the secure trapdoor for a given search query. Finally, t is the user defined search threshold.

3.2. Problem description

We consider the following setting: a data owner who owns a private collection of n textual documents $D = \{D_1, ..., D_n\}$. He intend to upload his private collection into a professional yet untrusted server, which provides data storage and computing services with an efficient cost. Before outsourcing, data owner generates a searchable index SI and uploads it together with the encrypted version of the encrypted documents C to the remote server. During search time, authorized users provide multi-keyword search request kw and use secret keys to construct a valid trapdoor T. Once receiving T, the server scans the searchable index SI to find the most similar encrypted documents that match the provided trapdoor. Beyond what data owner allows to leak, the security guarantee prevents the server from employing the outsourced data set and the provided trapdoors to get additional information about the underline collection and the search keywords.

3.3. Security definition

The objective of any SSE scheme is to protect: (1) document collection privacy, (2) user query privacy, and (3) searchable index privacy. Document collection privacy is protected using encryption before outsourcing. Interestingly, documents are encrypted by AES which supports PCPA (pseudo-randomness against chosen plaintext attack) security notion [6]. Such a notion guarantees that the ciphertext is indistinguishable from truly random string. User queries and document indices are transformed into binary vectors using secret keys. Such vectors hide both the number and content of the underline search keywords. The majority of the current SSE schemes define a leakage function, which describes precisely what is allowed to be leaked for the adversary server. We list some of the security definitions [6].

Definition 1: History (H_q) represents the document collection D and the set of search queries Q. Such issue represents the sensitive information that should not be leaked to the server. Formally, H_q= (D, Q). Definition 2: Access pattern ($A(Q_i)$) includes the collection of document identifiers $DB(Q_i)$ that satisfy the provided search query Q_i. Formally, $A(Q_i) = DB(Q_i)$. Definition 3: Search pattern (P) determines the repeated search queries, where it is determined by collecting the repeated queries. It is formulated as a square $q{*}q$ matrix, where $P(i, j) = 1$ iff $Q_i = Q_j$, $\forall i, j = 1, \ldots, q$. Definition 4: Trace ($Tr(H_q)$) it is the leakage function. Given the history H_q, trace is defined as: $Tr(H_q) = \{(id(C_1),\ldots, id(C_n)), (|C_1|,\ldots,|C_n|), A(H_q), P(H_q)\}$. Definition 5: View ($V(H_q)$) represents the actual information that the adversary server can see during the execution of the SSE scheme. Formally, $V(H_q) = \{(id(C_1),\ldots, id(C_n)), SI, C, Q\}$. Definition 6: Adaptive semantic security: SSE is adaptive semantic secure if given $Tr(H_q)$, there exists S algorithm that can simulates $V(H_q)$ with probability $1-\epsilon$ for all probabilistic polynomial time (PPT) adversaries, where ϵ is a negligible probability. Our scheme assumes the cloud server to be semi-honest party, where it follows exactly the construction steps, while try to analyze its accessed data to get additional information beyond what is allowed to learn.

4. THE PROPOSED SCHEME

We adopt a model of two-parties: the first one is the client (data owner), and the second one is the server. The client constructs an encrypted index, and outsources it along with the encrypted documents to the server, who provides large storage and responds to the search queries of the client. The proposed scheme is composed of five polynomial-time algorithms: Π=(`Gen`, `IndexBuilding`, `Makestrapdoor`, `Search`, `Dec`).

1. $(KS, KD) \leftarrow$`Gen`(1^{λ}): this algorithm is run by data owner. It takes the security parameter λ and returns two secret keys KS and KD.
2. $(SI, C) \leftarrow$ `IndexBuilding`(D, KS, KD): this algorithm is run by the data owner to generate the searchable index SI and the encrypted collection C, given the data collection D, and the secret keys KD and KS.
3. $T \leftarrow$ `Makestrapdoor`(kw, KS): Given the keyword set kw, data owner runs this algorithm to generate the secret trapdoor T utilizing the secret key KS.
4. $R \leftarrow$ `Search`(T, SI): Once receiving search trapdoor T, cloud server compares it with all of SI entries to find the list R of the most similar documents.
5. $D_i \leftarrow$ `Dec`(C_i, KD): given the secret key KD, data user runs this algorithm to get the plaintext form of document C_i.

4.1. Index building

In this work, a direct index [25] SI_i is generated for each document D_i. Under such a setting, search time is linear to the number of document collection files. The process of index generation includes three operations: keyword set extraction, document index construction, and trapdoor generation.

Keyword set extraction: Given the document D_i, data owner runs a preprocessing step to refine the keyword set of that document. Such step includes some techniques that are borrowed from information retrieval systems. Such techniques includes: lower case conversion, tokenization, removing stop words, and stemming [26]. Stemming converts the different forms of the same word into their unique stem. For example, the words talks, talking, and talked are all converted to the word talk. This process increases the probability of finding documents that have the provided keyword even in different forms. After refining the keyword set, the occurrence tf_j of each keyword w_j within D_i is computed. At the end, document D_i is represented as high dimensional vector: $D_i = \{(w_1, tf_1),\ldots,(w_m, tf_m)\}$.

Document index construction: Given the document keywords and their corresponding occurrences, we need to encode them together into a small fingerprint of fixed size l (usually l=64). To do so, we apply the Simhash function [27], which reduces high dimensional vectors such that two similar textual inputs will produce two fingerprints of a minimum Hamming distance. The fingerprint T of the document $D_i = \{(w_1, tf_1),\ldots,(w_m, tf_m)\}$ is generated as follows: initialize l zero counters $< s_1, ..., s_l >$. Each keyword $w_i \in D_i$ is hashed by SHA-1. If the bit j ($j \in \{1, ..., l\}$) equals 1, then its corresponding counter s_j is incremented by tf_i. Otherwise s_j is decremented by tf_i. After the processing of all document keywords, the counter value s_j ($j \in \{1, .., l\}$) is set to 1 if it is positive. Otherwise, it is set to 0. The fingerprint is the final binary bits of $< s_0, ..., s_l >$.

Trapdoor generation: Fingerprints are generated using SHA-1, which is a public function. However, using such a function raises security risks. This is because adversary server can guess search keywords with brute force attack, where it tries to compute the fingerprint for several inputs until it finds a collision. To solve this problem, we generate a trapdoor for each fingerprint using key based hashing function such as HMAC, where keywords are hashed using a secret key. This key will make the brute force attack a hard job. HMAC$_{KS}$:$\{0,1\}^* \rightarrow \{0,1\}^l$ [28] is a popular message authentication code that uses a secret key KS to hash each keyword of the given document. Other steps of Simhash function still without changing. Algorithm 1 illustrates index construction. Trapdoors of all document collection will be the final index that will be uploaded to the server.

Algorithm 1: Index building

```
Input  : Textual collection $D = \{D_1, D_2, \ldots, D_n\}$, secret keys: $KS$, and $KD$, fingerprint size $l$.
Output: Searchable index $SI$, encrypted collection $C$.
for $i \leftarrow 1$ to $n$ do
    $C_i \leftarrow Enc_{KD}(D_i)$;
    $\{(w_1, tf_1), (w_2, tf_2), , \ldots, (w_m, tf_m)\} \leftarrow$ PreProcessing $(D_i)$;
    Initialize zero vector $< s_1 \ldots s_l >$ ;
    for $j \leftarrow 1$ to $m$ do
        MAC $\leftarrow HMAC_{KS}(w_j)$;
        for $u \leftarrow 1$ to $l$ do
            if MAC$_u = 0$ then
                $s_u = s_u - tf_j$
            else
                $s_u = s_u + tf_j$
    for $u \leftarrow 1$ to $l$ do
        if $s_u \leq 0$ then
            $SI_i[u] = 0$
        else
            $SI_i[u] = 1$
    $SI = \{SI_1, ..., SI_n\}$;
    $C = \{C_1, ..., C_n\}$ ;
    Return $SI, C$;
```

4.2. Trapdoor construction

Given the search keyword set $kw = \{w_1, w_2, \ldots, w_s\}$, data user utilizes the secret key KS, which is obtained from the data owner via a secret channel, to generate Simhash fingerprint T for such a set. Trapdoor is constructed in the same way of document index generation. Note that the trapdoor size is l, which is unrelated to the number of keywords s, so s will be hidden from the adversary server. To construct a trapdoor, data user performs only hash functions and few bitwise comparisons.

4.3. Privacy preserving search

Cloud server evaluates the Hamming distance between the trapdoor T and each document index SI_i, $i = \{1, ..., n\}$. Particularly, he evaluates the XOR operation between two binary vectors. Similarity scores of two binary vectors are determined by identifying the number of matched bits. To control the number of returned results, users provide a threshold value t, such that only the encrypted files of the top-t minimum scores are returned. The search operation cost is illustrated as follows: cloud server needs to perform a binary comparison of l-bit fingerprint with n entries, then it sorts the similarity scores, and selects only the t-minimum scores. Algorithm 2 describes the steps of server search operation. Once receiving the t encrypted documents that have the best matching scores, data user utilizes the secret key KD to decrypt them.

Algorithm 2: Privacy preserving search

```
Input  : Simhash fingerprint $T$, threshold $t$, Searchable index $SI$, and encrypted collection $C$.
Output: encrypted documents of the top-$t$ similarity scores.
for $i \leftarrow 1$ to $n$ do
    XVect $\leftarrow$ XOR $(SI_i, T)$ ;
    Scores[$i$] $\leftarrow \sum_{j=1}^{l}$ XVect[$j$];
Sort_score $\leftarrow$ Sort (Scores);
Index $\leftarrow$ Minimum$_t$ (Sort_score);
for $j \leftarrow 1$ to $t$ do
    R$_j \leftarrow C$(Index[$j$]);
Return R;
```

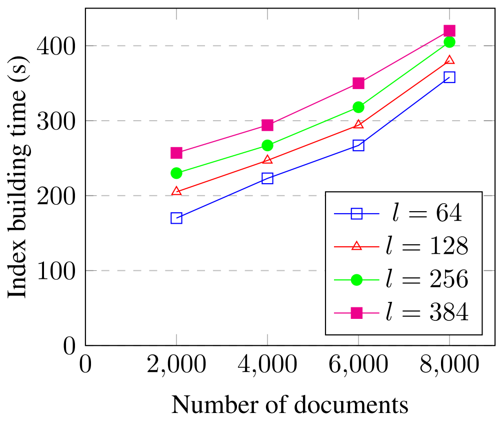
5. SECURITY DISCUSSION

We employ the real vs ideal games approach to prove the security of our proposed scheme according to Definition 6. In the real game, the client behaves as in the proposed scheme, while in the ideal game, we builds a simulator who employs the leaked information to simulate the behavior of the client towards the server. If there is no PPT distinguisher that can distinguish the behavior of the two games, then the scheme is considered a secure. The intuition of this approach is: given user private inputs H_q, accessible information to the server $V(H_q)$, and the leakage function $Tr(H_q)$, then if there exists a simulator S that employs $Tr(H_q)$ to simulate $V(H_q)$ such that no PPT distinguisher can distinguish $V(H_q)$ from the simulated one, then the leakage is not useful and hence the security is ensured. This is because $Tr(H_q)$ does not help the adversary to learn any useful information from $V(H_q)$. Formally, let the original view $V(H_q) = V(H_q) = \{(id(C_1), \ldots, id(C_n), SI, C, Q\}$, and the trace is $Tr(H_q) = \{(id(C_1), \ldots, id(C_n)), (|C_1|, \ldots, |C_n|), A(H_q), P(H_q)\}$. Let $V^*(H_q)$ be the simulated view, which is defined as: $V^*(H_q) = \{(id^*(C_1), \ldots, id^*(C_n)), SI^*, C^*, Q^*\}$. Our scheme is considered to be secure if $V^*(H_q)$ is indistinguishable from $V(H_q)$. Document identifiers $id(C_j)$ of $V(H_q)$ are available in $Tr(H_q)$. Thus S can simulate them simply by setting $id^*(C_j) = id(C_j)$. Simulator knows from $Tr(H_q)$ the number of outsourced documents and the real size of each one. Thus, it can create a simulated version C_j^* for each document C_j by encrypting a random stream of size $|C_j|$. This behavior cannot be distinguished by any PPT distinguisher due to the security of PCPA encryption method. Similarly, S simulates SI by generating n random binary vectors SI_j^* of size l. SI_j can not be distinguished from SI_j^* since it is generated using a secure hash function HMAC. Now, S simulates the q successive queries $Q = \{Q_1, Q_2, \ldots, Q_q\}$ of $V(H_q)$ as follows: if Q_j was not issued before, as indicated by $A(H_q)$ of $Tr(H_q)$, then it constructs a random Q_j^*. Otherwise it sets $Q_j^* = Q_p^*$, where p is previous index for Q_j^*. Note that $\forall i, Q_i^*$ is indistinguishable from Q_i. We explain how S can simulate $V(H_q)$. Hence, our proposed scheme is secure.

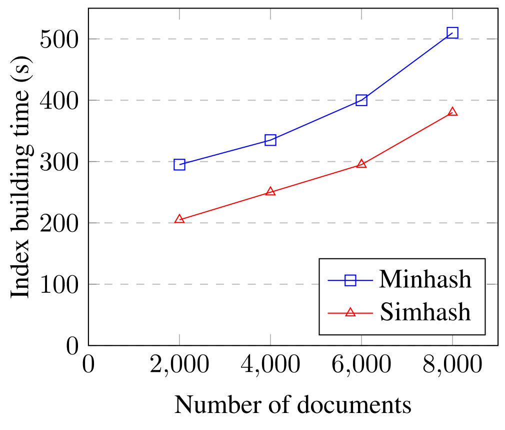
6. EXPERIMENTAL EVALUATION

We demonstrate the efficiency and effectiveness of our proposed scheme by evaluating thorough experiments. The proposed scheme is evaluated by Java language using 64-bit Windows 10 operating system with Intel Core-i7 processor of 1.8 GHz. During the experiments, 8000 randomly selected files are from the RCV1 database [29], a list of 570 stop words are used, Porter algorithm [30] is used to stem the remaining words.

Index generation time is determined by the parameter l. Figure 1a shows the effect of l for variable collection sizes. It is obvious that long Simhash fingerprints require more processing time.



(a)



(b)

Figure 1. Index building time: (a) effect of l on index generation time and (b) comparison of our scheme with [18] in terms of index building time

Our scheme is compared with the work of [18] in terms of index generation time. Figure 1b demonstrates that the method of [18] requires more time than our work, since it uses heavy cryptographic primitives.

We used two evaluation metrics to show the effectiveness of our scheme: precision and recall. Precision $prec$ refers to the percentage of relevant documents from the overall returned documents, while recall rec

refers to the percentage of relevant documents retrieved among the total number of relevant documents. Let $kw = \{w_1, w_2, \ldots, w_s\}$ be the set of search keywords, $R(kw)$ be the set of retrieved documents and $R^*(kw)$ be the set of retrieved documents that include all keyword set kw, $DB(kw)$ be set of documents in the entire collection that contains all keyword set kw. Note $R^*(kw) \subseteq R(kw)$ and $R^*(kw) \subseteq DB(kw)$. We define the precision($Prec(kw)$), recall ($Rec(kw)$), average precision ($Avgprec(kw)$) and Average recall ($Avgrec(kw)$) as follows:

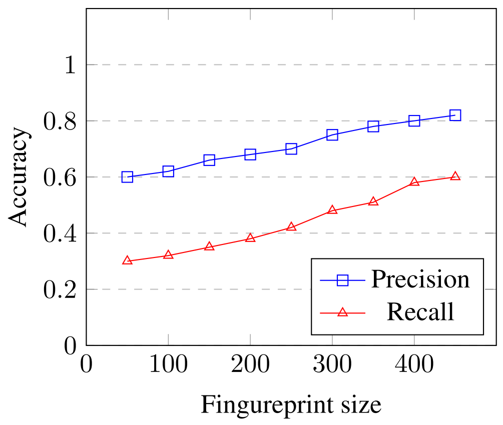
$$Prec(kw) = \frac{|R^*(kw)|}{|R(kw)|}, \tag{1}$$

$$Rec(kw) = \frac{|R^*(kw)|}{|DB(kw)|} \tag{2}$$

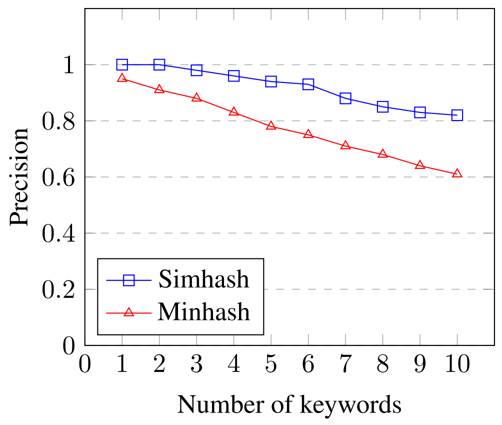
$$Avgprec(kw) = \sum_{i=1}^{q} \frac{prec(kw_i)}{q}, \tag{3}$$

$$Avgrec(kw) = \sum_{i=1}^{q} \frac{rec(kw_i)}{q} \tag{4}$$

Figure 2a shows the accuracy of results of our proposed scheme with variable Simhash size l. In this experiment, 150 search queries are provided with variable number of keywords and only the top 20 matching documents are returned.



(a)



(b)

Figure 2. Effectiveness evaluation: (a) Fingureprint size and (b) Number of keywords

In the next experiment, we evaluate how the number of keywords s in the search query can affect the result accuracy. Figure 2b shows the precision of results for different schemes with variable number of search keywords. For both schemes, precision decreases as the number of keywords increases from 1 to 10. This is because when the number of search keywords increased the non relevant retrieved documents will be accumulated, leading to lower precision. Notice that, the accuracy of our proposed scheme outperforms MinHash-based scheme, since it better integrates the term frequency of each keyword in the generated vector, whereas MinHash method ignores such valuable information.

7. CONCLUSION

In this paper, we proposed a practical scheme for a multi-keyword similarity search over encrypted data. The proposed scheme answers the multi-keyword queries and can handle the misspelled keywords. Documents are retrieved to the user according to specific ranking method. The scheme is constructed according to the setting of direct indexing, where a specific index is constructed for each document. Simhash function is employed to generate a small fixed size fingerprint for each document. Multi-keyword search queries are constructed in the same way, so the search operation is evaluated simply by performing Hamming distance. To illustrate the security of our scheme, a precis definition of security is presented along with its formal proof. Extensive experiments were conducted to show the practical value of our scheme. The proposed scheme shows high precision of 70% and recall of 85%. Our scheme requires 395 ms to generate the index for 8000 documents.

REFERENCES

[1] W. Hassan, T.-S. Chou, X. Li, P. Appiah-Kubi, and O. Tamer, "Latest trends, challenges and solutions in security in the era of cloud computing and software defined networks," *International Journal of Informatics and Communication Technology (IJ-ICT)*, vol. 8, no. 3, pp. 162–183, 2019, doi: 10.11591/ijict.v8i3.pp162-183.

[2] E. Goh, "Secure indexes," *IACR Cryptol. ePrint Arch.*, vol. 2003, p. 216, 2003. [Online]. Available: http://eprint.iacr.org/2003/216

[3] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Dynamic searchable encryption in very-large databases: Data structures and implementation," in *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014. [Online]. Available: https://www.ndss-symposium.org/ndss2014/dynamic-searchable-encryption-very-large-databases-data-structures-and-implementation

[4] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, ser. Lecture Notes in Computer Science, R. Canetti and J. A. Garay, Eds., vol. 8042. Springer, 2013, pp. 353–373, doi: 10.1007/978-3-642-40041-4_20.

[5] Y. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, ser. Lecture Notes in Computer Science, J. Ioannidis, A. D. Keromytis, and M. Yung, Eds., vol. 3531, 2005, pp. 442–455, doi: 10.1007/11496137_30.

[6] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *IACR Cryptol. ePrint Arch.*, vol. 2006, p. 210, 2006. [Online]. Available: http://eprint.iacr.org/2006/210

[7] 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, doi: 10.1109/INFCOM.2010.5462196.

[8] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *2012 IEEE 28th International Conference on Data Engineering*, 2012, pp. 1156-1167, doi: 10.1109/ICDE.2012.23.

[9] M. Strizhov and I. Ray, "Secure multi-keyword similarity search over encrypted cloud data supporting efficient multi-user setup," *Trans. Data Priv.*, vol. 9, no. 2, pp. 131–159, 2016, doi: 10.5555/2993206.2993208.

[10] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward private searchable symmetric encryption with optimized I/O efficiency," *IEEE Trans. Dependable Secur. Comput.*, vol. 17, no. 5, pp. 912–927, 2020, doi: 10.1109/TDSC.2018.2822294.

[11] J. G. Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "New constructions for forward and backward private symmetric searchable encryption," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM, 2018, pp. 1038–1055, doi: 10.1145/3243734.3243833.

[12] S. Chatterjee, S. K. P. Puria, and A. Shah, "Efficient backward private searchable encryption," *J. Comput. Secur.*, vol. 28, no. 2, pp. 229–267, 2020, doi: 10.3233/JCS-191322.

[13] I. Demertzis, J. G. Chamani, D. Papadopoulos, and C. Papamanthou, "Dynamic searchable encryption with small client storage." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1227, 2019. [Online]. Available: https://eprint.iacr.org/2019/1227

[14] G. Amjad, S. Kamara, and T. Moataz, "Forward and backward private searchable encryption with sgx," in *Proceedings of the 12th European Workshop on Systems Security*, 2019, pp. 1–6, doi: 10.1145/3301417.3312496.

[15] L. Bingjie, Z. Jun, and C. Zhenfu, "A multi-user forward secure dynamic symmetric searchable encryption with enhanced security," *Journal of Computer Research and Development*, vol. 57, no. 10, p. 2104, 2020.

[16] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222-233, Jan. 2014, doi: 10.1109/TPDS.2013.45.

[17] C. Örencik and E. Savas, "An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking," *Distributed Parallel Databases*, vol. 32, no. 1, pp. 119–160, 2014, doi: 10.1007/s10619-013-7123-9.

[18] C. Örencik, M. Kantarcioglu, and E. Savas, “A practical and secure multi-keyword search method over encrypted cloud data,” in *2013 IEEE Sixth International Conference on Cloud Computing*, 2013, pp. 390-397, doi: 10.1109/CLOUD.2013.18.

[19] C. Örencik, A. Selcuk, E. Savas, and M. Kantarcioglu, “Multi-keyword search over encrypted data with scoring and search pattern obfuscation,” *Int. J. Inf. Sec.*, vol. 15, no. 3, pp. 251–269, 2016, doi: 10.1007/s10207-015-0294-9.

[20] F. Baldimtsi and O. Ohrimenko, “Sorting and searching behind the curtain,” in *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, ser. Lecture Notes in Computer Science, R. Böhme and T. Okamoto, Eds., vol. 8975. Springer, 2015, pp. 127–146, doi: 10.1007/978-3-662-47854-7_8.

[21] Z. Xia, X. Wang, X. Sun, and Q. Wang, “A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340-352, 1 Feb. 2016, doi: 10.1109/TPDS.2015.2401003.

[22] C. Guo, R. Zhuang, C. Chang, and Q. Yuan, “Dynamic multi-keyword ranked search based on bloom filter over encrypted cloud data,” *IEEE Access*, vol. 7, pp. 35826-35837, 2019, doi: 10.1109/ACCESS.2019.2904763.

[23] B. Wang, W. Song, W. Lou, and Y. T. Hou, “Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee,” in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 2092-2100, doi: 10.1109/INFOCOM.2015.7218594.

[24] K. Pushpa Rani, L. Lakshmi, C. Sabitha, B. Dhana Lakshmi, and S. Sreeja, “Top-k search scheme on encrypted data in cloud,” *International Journal of Advances in Applied Sciences (IJAAS)*, vol. 9, no. 1, pp. 67–69, 2020, doi: 10.11591/ijaas.v9.i1.pp67-69.

[25] G. S. Poh, J. Chin, W. Yau, K. R. Choo, and M. S. Mohamad, “Searchable symmetric encryption: Designs and challenges,” *ACM Comput. Surv.*, vol. 50, no. 3, pp. 1-37, 2017, doi: 10.1145/3064005.

[26] M. 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,” *Nat. Lang. Eng.*, vol. 16, no. 1, pp. 100–103, 2010, doi: 10.1017/S1351324909005129.

[27] M. Charikar, “Similarity estimation techniques from rounding algorithms,” in *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, J. H. Reif, Ed. ACM, 2002, pp. 380–388, doi: 10.1145/509907.509965.

[28] M. Bellare, R. Canetti, and H. Krawczyk, “Keying hash functions for message authentication,” in *Advances in Cryptology - CRYPTO ’96*, N. Koblitz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 1-15, doi: 10.1007/3-540-68697-5_1.

[29] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, “Rcv1: A new benchmark collection for text categorization research,” *J. Mach. Learn. Res.*, vol. 5, p. 361–397, Dec. 2004.

[30] M. F. Porter, “An algorithm for suffix stripping,” *Program: electronic library and information systems*, vol. 40, no. 3, pp. 211-218, 2006, doi: 10.1108/00330330610681286.