Journal of Basrah Researches ((Sciences)) Vol. (47). No. 1 (2021)

Available online at: https://www.iasj.net/iasj/journal/260/issues

ISSN –1817 –2695

Received date: 7-8-2021 Accepted date: 10-11-2021 Available online date: 31-12-2021

Multi-keyword search over encrypted data with security proof

Hussein M. Mohammed, Ayad I. Abdulsada*,
Department of Computer science, Education College for Pure Sciences,
University of Basrah, Basrah, 61004, Iraq

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

Abstract

Cloud computing is an information technology that includes efficient servers provide storage and computing services with an efficient costs. Users still hesitate to outsource their private data to such untrusted servers due to privacy issues. Encryption is usually applied for data before being uploaded. However, such a scheme disables most data processing. Searchable encryption SE schemes are developed to allow for searching the encrypted data. Unfortunately, such schemes are commonly deal with only a single keyword queries. In this paper, we provide a new scheme that allows to search with multi-keyword queries and provide a formal security proof. Our proposed scheme incorporates two servers to hide the search pattern and access pattern. Experiments utilizing real-world data demonstrate that our proposed scheme is practically efficient, secure, and holds high accuracy.

KEYWORDS: Cloud computing, privacy preserving, searchable encryption, multi-keyword search.

I. INTRODUCTION

Cloud computing is a new technology that employs efficient servers with huge computational power and storage to process and store the uploaded data with low price. Users and companies are encouraged to outsource their data into such servers. However, privacy raises significant concerns about outsourced data, especially with private data like patient documents, photos, bank credits, etc. Therefore, users tend to encrypt their data and upload the encrypted data for the untrusted servers. Such a treatment, despite its protection, makes it impossible to perform the basic operations for data processing. Hence, cryptographers have developed searchable encryption SE scheme that allows for searching the encrypted data without being decrypted.

Consider we have a text file collection. SE schemes extract a searchable index from that collection and encrypt both the real collection and the index and upload the results to the cloud servers. When the data owner intents to search his data, he/she generates a search trapdoor and sends it to the server. The latter runs a search protocol to find the matched entries and returns them back to the user. The efficient SE scheme usually leaks the access and search pattern for efficiency purpose. The search pattern indicates whether the underlying keyword has been searched before or not, while the access pattern refers the search results. The study of [1] shows the security ramifications of these leaked information and show how such information can be used to infer the entire file collection.

The majority of existing SE schemes are designed to answer only a single keyword queries (e.g., [2-5]) that return many results. Multi-keyword SE schemes (e.g., [6-8]) can handle search queries with several keywords. Hence, search results are filtered using the relevance scores. However, none of the above-mentioned SE schemes can able to deal with misspelled keywords. Actually, these schemes are designed only to deal with exact keywords. Few schemes in the literature are designed to serve fuzzy search (e.g., [9-12]), where a fuzzy set is extracted and stored for each keyword. For example, within edit distance 1 the fuzzy set of "bus" is {bus, *bus, b*s, ba*}. Notice that such a treatment can handle single keyword at high storage cost.

In [13], the authors proposed the first secure scheme that handle multi-keyword similarity search. However, this scheme leaks the access and the search patterns, which have been proved to raise serious security problems [1, 14, 15]. Furthermore, this scheme does not incorporate the term frequency of the indexing keywords during document representation. Recently, Liu et. al [16] have proposed an effective scheme that support fuzzy and semantic SE, they handle fuzzy search by utilizing fingerprint generation algorithm and employing Hamming distance. Guo et. al [17] have designed a secure scheme that support similarity search, they utilize a novel way to improve security (where

two-server setting and a random index are exploited to conceal the query distribution), The basic idea of Guo′s scheme that represents to combine SSE with LSH.

In this paper, we proposed a multi-keyword searchable encryption scheme that recovers the problems of [13]. Specifically, our scheme hides the search and access patterns. To achieve this goal, we extract a numerical vector from each textual file and encrypt that vector by adopting the homomorphic encryption. Our scheme employs two-servers to hide the search pattern leakage.

TABLE 1.
REPRESENTS COMPERATIONS BETWEEN THE SCHEME OF [13] AND OUR SNHEME

Features	Wang et al. [13]	Our Scheme
Search	Fuzzy	Fuzzy & Similarity
Keyword Stripping	Unsupported	Support
Ranking	Unsupported	Support
Servers	One Server	Two Server
Search & Access patterns	Revealed	Hide

The contributions can be illustrated as follows:

1) We introduce a searchable encryption scheme that supports multi-keyword similarity queries.
2) The access, similarity and search patterns are protected
3) We provide a formal security proof.

The paper is organized by the following sections: Section II reports the most relevant works. Section III presents the problem description. Section IV shows the proposed scheme. Section V provides the security proof. Section VI concludes the paper.

II. RELATED WORKS

Song et al. [2] proposed the first SE scheme that searches the encrypted files directly. Goh [3] enhances the search cost by using a forward index for each file. Curtmola et al.'s [4] utilized the reverse index to search the encrypted files with sublinear cost. Chang et al. [5] enhances the security definition of [3]. However, these schemes support only queries with single keywords.

The first multi-keyword SE scheme is due to Cao et al. [18], where each file is transformed into very long numerical vector. Sun et al. [19] improve the work of [18] by using a tree data structure to maintain the searchable index. However, these schemes support exact multi-keyword search. Li et al. [9] introduce the first SE scheme that handles fuzzy search by using a wildcard technique to generate a fuzzy set for each keyword. The set of each keyword includes all the possible modifications of that keyword. The storage cost of [9] was reduced in [10] where the gram-based technique is used to extract smaller sets. Kuzu et al. [12] build a revers index for the entire collection by indexing the minhash terms. However, such schemes can search only with single keyword queries. The schemes of [13, 20] to improve the search efficiency by using Bloom filter to describe files. Fu et al. [21] enhance the accuracy of [13], where the keywords of file are transformed into uni-gram vectors that are mapped into single file vector by using LSH hashing method. Liu et. al [16] have proposed an effective scheme that support fuzzy and semantic SE, they handle fuzzy search by utilizing fingerprint generation algorithm and employing Hamming distance. Guo et. al [17] have designed a secure scheme that support similarity search, they utilize a novel way to improve security (where two-server setting and a random index are exploited to conceal the query distribution), The basic idea of Guo′s scheme that represents to combine SSE with LSH.

TABLE 2.
REPRESENTS COMPERATIONS BETWEEN THE PREVIOUS SCHEMES AND OUR SNHEME

Features	Single-keyword	Multi-keyword	Exact keyword	Fuzzy keyword	Similarity keyword	Result ranking	Reveal search & access patterns	Hide search & access patterns
Song et. al [2]	√		√				√	
Goh [3]	√		√				√	
Curtmola et. al [4]	√		√				√	
Li et. al [9]	√			√			√	
Li et. al [10]	√			√			√	
Kuzu et. al [12]	√				√	√		√
Cao et. al [18]		√	√			√	√	
Wang et. al [13]		√		√			√	
Fu et. al [21]		√		√		√	√	
Our scheme	√	√	√	√	√	√		√

III. PROBLEM DESCRIPTION

Figure 1 illustrates our scheme which includes four parties: data owner, client, file server, and search server. We assume that the two servers are non-colluding servers and they are used to break the relation between search queries and their corresponding results.

The *data owner* (DO) owns n textual files $F = (f_1, f_2, \dots, f_n)$. DO wants to outsource F to the cloud server since he does not have sufficient storage cost. To protect the privacy of his data, DO applies encryption process on F to get the encryption collection $C = (c_1, c_2, \dots, c_n)$ and outsources C to the cloud server. To enable an efficient search DO extracts searchable index I from F and outsources it also to cloud server. The *client* (CL): has a multi-keywords query q. To submit this query, CL first constructs a searchable token T from q by using a private key, then sends T to the cloud server. The *search server* (SS)

is used to store searchable index I and receives the search tokens from the clients. The *file server* (FS) stores the encrypted collection C and works together with SS to generate the similarity score for each file. Finally, FS will receive the encrypted scores. It decrypts them and returns them back to CL, who maps each score with its real file and selects only the best k matching files.

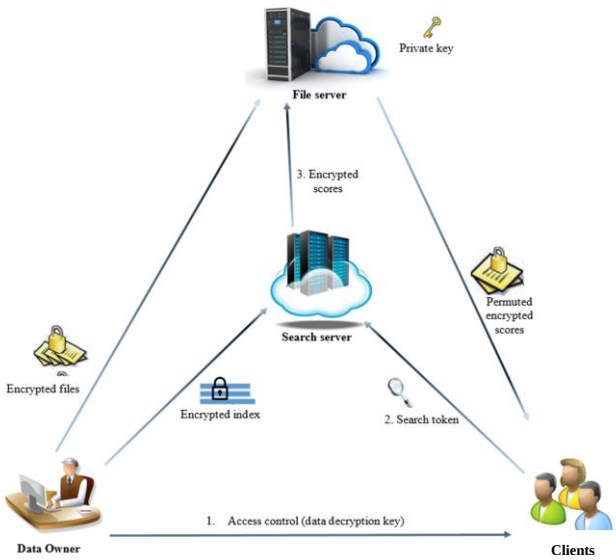

Fig. 1: The system model

With respect to security model, we employ the "honest-but-curious" as in [3, 9, 22], where all the involved parties are assumed to perform precisely the operations of the search protocol but they want to get additional private information. The security requirements are defined as follows:

1. *File privacy*: file contents should be hidden to the cloud server.
2. *Index confidentiality and query privacy*: Files and queries should be described as indexes that protect the underlying keywords.
3. *Search pattern protection:* the fact that two identical queries are presented should not be revealed.
4. *Access pattern protection*: The cloud server should not be able to correlate the search results with the provided search trapdoors.
5. *Similarity pattern protection*: The cloud server should not be able to detect the common keywords between two search queries.

TABLE 3.
COMMON NOTATIONS

Symbols	Descriptions
W	The keyword set, $W = \{w_1, \dots, w_m\}$
m	The number of keywords
F	The file collection, $F = \{f_1, \dots, f_n\}$
n	The number of files in F
C	The encrypted file collection, $C = \{c_1, \dots, c_n\}$
I	Searchable index
pk_p	Public key for encryption
sk_p	Private key for decryption
q_i	i[th] query in the query set.
T	The search trapdoor of keywords
k	Number of returned files
$BF(f)$	The Bloom filter of file f
$[\![BF(f)]\!]$	The encrypted Bloom filter
γ	Bloom filter size
l	Number of hash functions
PPT	Probabilistic polynomial time

IV. The proposed scheme

Our scheme extracts the keyword set of each file and represent this set as a γ-bit numerical vector (NV). To enhance the ability of recovering the misspelled keywords, we apply Porter algorithm [23] to transform each keyword into its original form. To build the index NV, each keyword w is transformed into bi-gram vector. This vector is represented as a 26^2-bit long vector. If a given bigram (e.g. "ab") is present in the bigram set of a given keyword, the relevant entry of the bigram vector will be 1. Otherwise, it will be 0. This representation can recover the misspelled keywords.

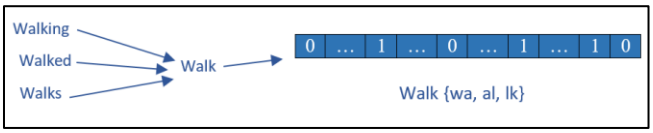

Fig. 2: Illustrates keyword representation

Bi-gram vectors are hashed several times by using LSH hash functions to find the locations of NV that will store the value 1. LSH functions are used since they have the ability to hash the similar keywords into the identical locations. For enhancing the ranking functionality, the term frequency of each keyword is stored in the NV entries. To protect NV we utilize Paillier encryption [24] to encrypt its entries before uploading NV to the cloud server. Search queries are considered as textual files. Thus, their index is extracted and encrypted in a similar way. The similarity score between two encrypted indexes is produced by evaluating secure inner product protocol by using the two non-colluding servers. We use Paillier cryptosystem since it supports homomorphic

properties that enable to add two encrypted numbers without decryption. Observe that, the inner product between two numerical vectors is the addition of the sub-multiplications of all entries in the involved vectors.

The secure multiplication of two numbers is denoted by $[\![xy]\!] \; mod \; N^2$. However, Paillier cryptosystem does not providing the multiply property, so we need to find a solution for this problem. Our treatment for solving this problem requires two servers: SS and FS. FS has access to the Paillier private key. SS masks the two multiplication numbers by using random numbers and moves the results to FS who decrypts the received numbers, multiplies the masked number, and then encrypts the result. Finally, SS exploits the homomorphic property to subtract the random numbers and get the actual encrypted multiplication. Algorithm 1 shows the process of multiplying two encrypted numbers.

Algorithm 1 Secure Multiplication ($[\![x.y]\!]$)

SS:
Input: x, y
Output: x', y'
pick random $n_1, n_2 \; mod \; N$
$x' = [\![x]\!] \times [\![n_1]\!] = [\![x + n_1]\!]$
$y' = [\![y]\!] \times [\![n_2]\!] = [\![y + n_2]\!]$
send x', y' to FS
FS:
Input: x', y'
Output: h'
$Dec(x') = x + n_1$
$Dec(y') = y + n_2$
$h = Dec(x') \times Dec(y') \; mod \; N$
$h = xy + yn_1 + xn_2 + n_1 n_2 \; mod \; N$
$h' = [\![h]\!]$
send h' to SS
SS:
Input: h'
Output: $[\![x.y]\!]$
$[\![x.y]\!] = h' \times [\![x]\!]^{N-n_2} \times [\![y]\!]^{N-n_2} \times [\![r_1 r_2]\!]^{N-1} \; mod \; N^2$

Algorithm 2 is used to aggregate the partial products of all sub results that are obtained from Algorithm 1. Specifically, Algorithm 2 computes in a secure way the dot product by utilizing Paillier's addition homomorphic property. To conceal the relation between files and their similarity scores, SS applies a permutation function π (that is provided by the search user) on the encrypted scores. The permutation function is changed after each search, so it is impossible for the search server to conduct any inference attack.

Algorithm 2 Secure similarity Search

SS:
Input: $[\![BF(f_1)]\!], \ldots, [\![BF(f_n)]\!]$: n encrypted Bloom filters
- $[\![BF(q))]\!]$: query Bloom filter, π : random permutation
- for each $[\![BF(f_i)]\!] \in BF$ do
  - $[\![dot]\!] = [\![0]\!]$
  - for $j = 1 \; to \; \lambda$ do
    $[\![dot]\!] = [\![dot]\!].$
    $\left[\!\left[secure_{dot_{prduct}}(BF(f_i), BF(q)) \right]\!\right]$
  - $[\![Scores[i]]\!] = [\![dot]\!]$
- Permute $\{ [\![Scores[1]]\!], \ldots, [\![Scores[n]]\!] \}$ with random permutation π and send the result to FS

FS:
Input: permuted encrypted scores
- Decrypt Scores and send the results to the user

CL:
- Perform the inverse permutation π on the similarity scores to associate them with their actual identifiers

Sort the scores to get file identifiers with maximum dot product.

VI. Security Analysis

In this section, we'll analyze the security of our scheme. Before the proof, we will illustrate some notations and definitions. Let F be a file set $F = \{f_1, \ldots, f_n\}$, and Q is a set of queries $Q = \{q_1, \ldots, q_m\}$ that are submitted by the search user, $T = \{T_1, \ldots, T_m\}$ be the search trapdoors for Q, R be a file identifiers that match the search query, }, I is a secure searchable index. PPT stands for probabilistic polynomial time.

Definition 1 History H: is the sensitive information that need to be protected, $H = \{F, Q\}$.
Definition 2 Search Pattern (SP): the search pattern of the history $H(F, Q)$ is a binary matrix SP(H) such that for $1 \leq i, j \leq m$, $SP[i, j] = 1$ if $q_i = q_j$ and $SP[i, j] = 0$ otherwise.
Definition 3 Access Pattern (AP): the access pattern of the history $H(F, Q)$, is defined as

$$AP(H) = \{R(q_1), \ldots, R(q_m)\}.$$

Definition 4 Similarity Pattern (SIMP): Let we have two search queries: q_1 and q_2, the SSE scheme achieves SIMP when no efficient adversary can recognize the existing or not of a common keywords between q_1 and q_2.

Definition 5 View of history V (H): the view is the only thing the cloud server can see and it includes

$$V(H) = \{id(c_1), \dots, id(c_n), I, C, T\}$$

Definition 6 Trace of history Tr(H)**:** A trace of H is the information that is allowed to be leaked to the cloud server. It is defined as:

$$Tr(H) = \{|c_1|, \dots, |c_n|, id(c_1), \dots, id(c_n), |I|, SP(H), AP(H)\}$$

Definition 7 Adaptive Semantic Security. The SSE scheme for similarity is described as adaptively semantically secure, if for all PPT adversaries A there exists a PPT simulator S, such that S can use the trace to construct the view of the history with high probability. Interestingly, this method guarantees that all the information that is available to the cloud server could be obtained from the allowed leakage only. Therefore, similarity SSE scheme that satisfies this definition leaks no information beyond the trace. More formally, similarity SSE scheme satisfies the security definition if for all simulators S we can write

$$Pr[D(V(H)) = 1] - Pr[D(S(Tr(H)) = 1)] < \frac{1}{g(r)}$$

Where D is a PPT distinguisher, and g(r) is polynomial for large r.

For ease of presentation, we take the following game:

Let we have SSE scheme that includes the following algorithms: Gen for key generation, Enc for files and index encryption , Trpdr for generating valid trapdoors from search queries, Search for searching the encrypted index, and Dec for decrypting the received files. Furthermore, let $k \in N$ be the security parameter, A be an adversary that conducts a consecutive steps: $A_0, \dots, A_q$, $m \in N$, and simulator S of $S_0, \dots, S_q$ steps, and suppose the following PPT games $Real^*_{SSE,A}(k)$ and $Sim^*_{SSE,A}(k)$:

$\mathbf{Real}^*_{SSE,A}(k)$

$K \leftarrow Gen(1^k)$

$(\boldsymbol{F}, st_A) \leftarrow A_0(1^k)$

$(I, \boldsymbol{c}) \leftarrow Enc_k(\boldsymbol{F})$

$(q_1, st_A) \leftarrow A_1(st_A, I, \boldsymbol{c})$

$t_1 \leftarrow Trpdr_k(q_1)$

$for\ 2 \le i \le m,$

$(q_i, st_A) \leftarrow A_i(st_A, I, c, t_1, \dots, t_{i-1})$

$t_i \leftarrow Trpdr_k(q_i)$

$let\ \boldsymbol{t} = (t_1, \dots, t_q)$

$output\ v = (I, \boldsymbol{c}, \boldsymbol{t})\ and\ st_A$

$\mathbf{Sim}^*_{SSE,A}(k)$

$(\boldsymbol{F}, st_A) \leftarrow A_0(1^k)$

$(I, \boldsymbol{c}, st_S) \leftarrow S_0(\tau(\boldsymbol{F}))$

$(w_1, st_A) \leftarrow A_1(st_A, I, \boldsymbol{c})$

$(t_1, st_S) \leftarrow S_1(st_S, \tau(\boldsymbol{F}, q_1))$

$for\ 2 \le i \le m,$

$(q_i, st_A) \leftarrow A_i(st_A, I, \boldsymbol{c}, t_1, \dots, t_{i-1})$

$(t_i, st_S) \leftarrow S_i(st_S, \tau(\boldsymbol{F}, q_1, \dots, q_i))$

$let\ \boldsymbol{t} = (t_1, \dots, t_q)$

$output\ v = (I, \boldsymbol{c}, \boldsymbol{t})\ and\ st_A$

We say that SSE is adaptively semantically secure, if there exists a PPT simulator S for all PPT adversaries A and for all PPT distinguishers D,

$$|Pr[D(v, st_A) = 1 : (v, st_A) \leftarrow Real^*_{SSE,A}(k)] - |Pr[D(v, st_A) = 1 : (v, st_A) \leftarrow Sim^*_{SSE,A}(k)]| \le negl(k)$$

Theorem 1: The proposed SSE scheme achieves Definition7.

Proof:

We will demonstrate that simulator S is able to simulate the real view V(H) of all possible histories H and generate simulated view VS(H) such that no PPT party can be able to distinguish them.

Let $V(H) = \{id(c_1), \dots, id(c_n), I, C, Q\}$ be the original view, and $Tr(H) = \{|c_1|, \dots, |c_n|, id(c_1), \dots, id(c_n), |I|\}$ be the legal leakage of H. Then, S produces $VS(H) = \{id^*(c_1), \dots, id^*(c_n), I^*, C^*, Q^*\}$ as follows:

- Simulation of file identifiers: the file identifier $id(c_i)$ is available in the trace thus S can easily set $id^*(c_i) = id(c_i)$
- Simulation of encrypted documents: S chooses n random values $\{C^*_1, \dots, C^*_n\}$. Since files are encrypted using PCPA-security encryption scheme [5], the outputs of such a scheme cannot be differentiated from random values. Thus, c^*_i and c_i are indistinguishable. Notice that $|c_i|$ is available in the trace so S can ensure that $|c_1| = |c^*_1|$.
- Simulation of the index: Notice that I is a set of encrypted vectors of λ elements. Simulator *S* constructs |I| vectors, such that $I^*[i]$ is a random vector, where $|I^*[i]| = \lambda$. Intuitively, $I^*[i]$ and I[i]

are indistinguishable since all elements of I[i] are semantically encrypted. So, I is indistinguishable from I^*.

- Simulation of trapdoors: S simulates trapdoors $\{T_{q_1}, \ldots, T_{q_m}\}$. Let $T_{q_i} = < BF(q_i)[1], \ldots, BF(q_i)[\lambda] >$ be a fixed vector trapdoor with λ elements. The simulator sets $T_i[j]^*$ to random value. Note that, no PPt party can distinguish $T_i[j]$ from $T_i[j]^*$ since $T_i[j]$ is the output of a Paillier encryption.

Since the components of $VR(H_n)$ and $V(H_n)$ are computationally indistinguishable, we can conclude that the provided scheme satisfies the security definition 7.
Now we discuss how our scheme meets the security requirements:

1) File privacy: The file content is encrypted by the data owner by utilizing the symmetric encryption algorithm (e.g. AES) before outsourcing it to the server. If the server doesn't obtain the key, the obtained ciphertext file cannot be decrypted. Thus file privacy is ensured.
2) Index privacy and Query privacy: Note that, in our scheme, each file or query is transformed into encrypted numerical vectors that conceal file contents and the number of keywords. Therefore, index privacy and query privacy are preserved
3) Search and similarity patterns: file and query vectors are encrypted using a semantically secure encryption procedure. This means that each element can't be distinguishable from a random number. So, search and similarity patterns are preserved.
4) Access pattern: the search server deals with only encrypted scores so cannot learn the access pattern. The file server work on permutated scores, thus it cannot learn the access pattern.

Now we have to prove the security of the secure multiplication protocol (Algorithm 1). Recall that such a protocol works under the setting of two-party computing, where two semi-honest servers are participated to complete the work. We need to prove that such a protocol achieves private computation. Again, we utilize the simulation approach, where the protocol is secure only when the view of any server can be simulated efficiently given only the inputs and outputs of that server. This means that the view doe not provide any useful information.

In Algorithm 1, the input of *SS* is two encrypted numbers $[[x]]$, $[[y]]$ and its output is $[[x.y]]$. The view of *SS* is $[[h]]$. The simulator knows Paillier's public key. Thus, it can simulate the real view of *SS* by encrypting the number zero to get $[[h']]$. Notice that $[[h]]$ and $[[h']]$ are indistiguishable since the semantic property of Paillier cryptosystem.

The inputs of FS are $x + n_1$ and $y + n_2$, and its output is $[[h]]$. The real view of FS is $[[x + n_1]]$ and $[[y + n_{12}]]$. The simulator uses the public key to simulate view of FS by encrypting $x + n_1$ and $y + n_2$.

VII. Experimental results

We test our scheme on a real-data collection from 1000 Wikipedia articles. In the used files we extract 4733 distinct keywords,. For LSH we use $l = 40$ hash functions, and fix $\gamma = 800$. We used a 64- Windows running on a PC of 1.6 GH Intel corei5 CPU, and 8 GB RAM. We compare our file description method two state of the art methods: Term vector of [25] and Simhash of [26].

Figure 3 shows the running time for building encrypted vectors for the all files with γ. It is clear to see that more time is needed for larger γ values

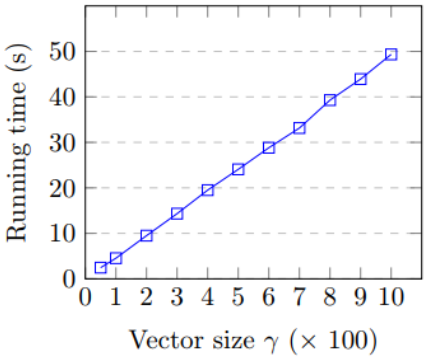

Fig. 3: Encrypted vector building time.

Figure 4 reports the CPU running time for searching the whole file collection. Notice that larger vectors require immediately more search time.

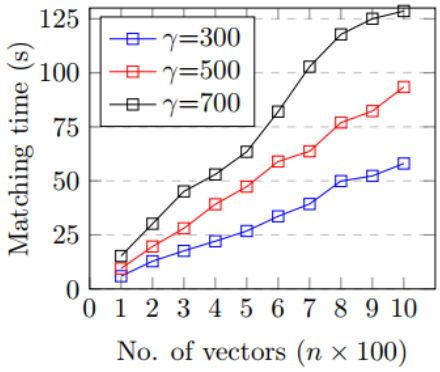

Fig. 4: Search time.

Figure 5 demonstrates the effect of fuzzy keywords on the precision of results. In this experiment we

provide query with misspelled keywords and returns only the best 6 files (i.e. k=6). Notice that our proposed scheme has the advantage over the other schemes to handle the misspelled keywords.

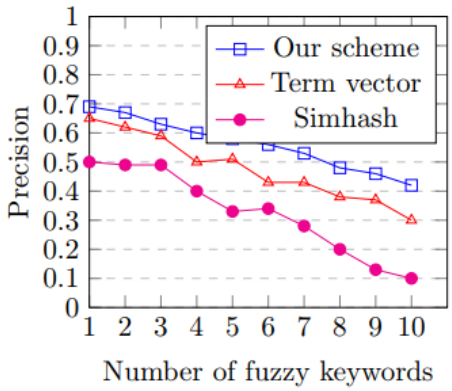

Fig. 5: handling misspelled keywords.

VIII. CONCLUSION

In this paper, we address the problem of multi-keyword similarity ranked search over the encrypted cloud data. We adopt the locality sensitive hashing functions and Bloom filters to represent files. Our scheme shows an enhanced security level, where access and search patterns are preserved. While the works of [13, 21] reveal these patterns. To do so, we utilized two servers. We provide security analyses and proof and test the practical value of our scheme via excremental results on real-world files.

REFERENCES

[1] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: ramification, attack and mitigation," in *Ndss*, 2012, vol. 20, p. 12: Citeseer.

[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] E.-J. J. I. C. e. A. Goh, "Secure indexes," vol. 2003, p. 216, 2003.

[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.

[5] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *International conference on applied cryptography and network security*, 2005, pp. 442-455: Springer.

[6] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *International conference on applied cryptography and network security*, 2004, pp. 31-45: Springer.

[7] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *International conference on information and communications security*, 2005, pp. 414-426: Springer.

[8] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of cryptography conference*, 2007, pp. 535-554: Springer.

[9] 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.

[10] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. J. I. C. e. A. Lou, "Enabling Efficient Fuzzy Keyword Search over Encrypted Data in Cloud Computing," vol. 2009, p. 593, 2009.

[11] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 451-459: IEEE.

[12] 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: IEEE.

[13] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, 2014, pp. 2112-2120: IEEE.

[14] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 668-679.

[15] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 707-720.

[16] G. Liu, G.Yang, S. Bai, Q. Zhou, and H. J. A. Dai, "FSSE: An effective fuzzy semantic searchable encryption scheme over encrypted cloud data" vol. 8,pp. 71893-71906, 2020.

[17] C. Guo, W. Liu, X. Liu, and Y. J. L. T. o. C. C. Zhang, "Secure Similarity Search over Encrypted Non-Uniform Datasets ", 2020.

[18] 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.

[19] 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.

[20] C.-M. Yu, C.-Y. Chen, and H.-C. J. I. s. j. Chao, "Privacy-preserving multikeyword similarity search over outsourced cloud data," vol. 11, no. 2, pp. 385-394, 2015.

[21] Z. Fu, X. Wu, C. Guan, X. Sun, K. J. I. T. o. I. F. Ren, and Security, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," vol. 11, no. 12, pp. 2706-2716, 2016.

[22] J. Wang, X. Yu, M. J. A. J. f. S. Zhao, and Engineering, "Privacy-preserving ranked multi-keyword fuzzy search on cloud encrypted data supporting range query," vol. 40, no. 8, pp. 2375-2388, 2015.

[23] M. F. J. P. Porter, "An algorithm for suffix stripping," 1980.

[24] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*, 1999, pp. 223-238: Springer.

[25] W. Jiang, M. Murugesan, C. Clifton, and L. Si, "Similar document detection with limited information disclosure," in *2008 IEEE 24th International Conference on Data Engineering*, 2008, pp. 735-743: IEEE.

[26] S. Buyrukbilen and S. Bakiras, "Secure similar document detection with simhash," in *Workshop on Secure Data Management*, 2013, pp. 61-75: Springer.