International Journal of Parallel Programming (2020) 48:470–495
https://doi.org/10.1007/s10766-018-0584-8

Countering Statistical Attacks in Cloud-Based Searchable Encryption

M. A. Manazir Ahsan[1] · Ihsan Ali[1] · Mohd Yamani Idna Bin Idris[1] · Muhammad Imran[2] · Muhammad Shoaib[2]

Received: 21 March 2018 / Accepted: 21 June 2018 / Published online: 30 June 2018

Abstract
Searchable encryption (SE) is appearing as a prominent solution in the intersection of privacy protection and efficient retrieval of data outsourced to cloud computing storage. While it preserves privacy by encrypting data, yet supports search operation without data leakage. Due to its applicability, many research communities have proposed different SE schemes under various security definitions with numerous customary features (i.e. multi keyword search, ranked search). However, by reason of multi-keyword ranked search, SE discloses encrypted document list corresponding to multiple (secure) query keywords (or trapdoor). Such disclosure of statistical information helps an attacker to analyze and deduce the content of the data. To counter statistical information leakage in SE, we propose a scheme referred to as Countering Statistical Attack in Cloud based Searchable Encryption (CSA-CSE) that resorts to randomness in all components of an SE. CSA-CSE adopts inverted index that is built with a hash digest of a pair of keywords. Unlike existing schemes, ranking factors (i.e. relevance scores) rank the documents and then they no longer exist in the secure index (neither in order preserving encrypted form). Query keywords are also garbled with randomness in order to hide actual query/result statistics. Our security analysis and experiment on request for comments database ensure the security and efficiency of CSA-CSE.

Keywords Cloud computing · Security · Searchable encryption · Statistical attack · Multi keyword ranked search

✉ Ihsan Ali
ihsanalichd@siswa.um.edu.my

Extended author information available on the last page of the article

1 Introduction

Cloud computing is a promising computing paradigm that brings a new era in information technology arena [1–4]. It provides on-demand and scalable computing and storage facilities to individuals or business customers. Low-cost resources and convenient payment policy (pay as you go) are attracting clientele rapidly. Similarly, cloud storages e.g. Amazon storage, Dropbox, Google Drive and so on are grabbing an increasing number of market share. Individual users store their data to the cloud and access it via network ubiquitously. Besides, the cloud exempts enterprises from maintaining computing/storage infrastructures as well as from managing human resources to look after them. At the same time, cloud storage assists to fight an emerging threat ransomware [5] by preserving data redundantly. However, cloud is not risk free and there exist some potential security and privacy concerns for cloud storage [6–9].

When users upload their data to the cloud, they lose the physical control of the data and cannot protect it from unauthorized access especially from the cloud service provider (CSP) itself. There are some prominent cyber accidents in the history, for example, Apple's iCloud leakage in 2014, Dropbox data privacy breach in 2016. Particularly from iCloud's leakage event, numerous Hollywood actresses' private photo got exposed and caused massive outcry [10]. To protect sensitive data, users encrypt it before outsourcing to the cloud. But, encryption converts the data into random text and obviates from search based utility. Definitely, downloading all the data and decrypting them locally before search operation is not a feasible solution because of massive communication and computation cost respectively. To resolve this problem, searchable encryption is an essential tool. Searchable encryption (SE) is a cryptosystem, in which encrypted text can be searched [11–13]. Confidentiality of data is preserved through encryption, yet search option is provided without data leakage. Depending on application scenario, SE can be subdivided into two major classes: data sharing and data outsourcing scenario. In data sharing, a sender sends data to a receiver via a gateway i.e., cloud server. Data sharing in symmetric key encryption requires complex key management [14] in contrast to asymmetric cryptography which can handle SE conveniently. Likewise, in data outsourcing, user outsources encrypted data to the cloud and then uses secure query keyword to search. Most of the solutions of SE for data outsourcing involve symmetric key cryptosystem for efficiency. This work focuses on searchable encryption in data outsourcing context.

In cloud data outsourcing scheme, the user generates an index with a collection of document IDs and each document ID is followed by a keyword set extracted from that document. Then user encrypts the documents using modern cryptosystem (such as RSA, AES or Salsa20) and transforms the index into secure index (SI) that contains secure searchable keywords. Figure 1 demonstrates the index and SI data structure. More precisely, Fig. 1a, b illustrates the index and SI respectively. User uploads the encrypted documents along with the SI to the cloud. During search operation, the user generates trapdoor for query keyword set and sends the trapdoor to the cloud. Trapdoor is cryptographically irreversible transformation of query keywords used to search similarity against SI. Cloud compares the trapdoor with the secure index to test similarity. Once the cloud server finds the similarity, it sends the relevant encrypted documents back to the user. This is typical interaction for searching on encrypted data

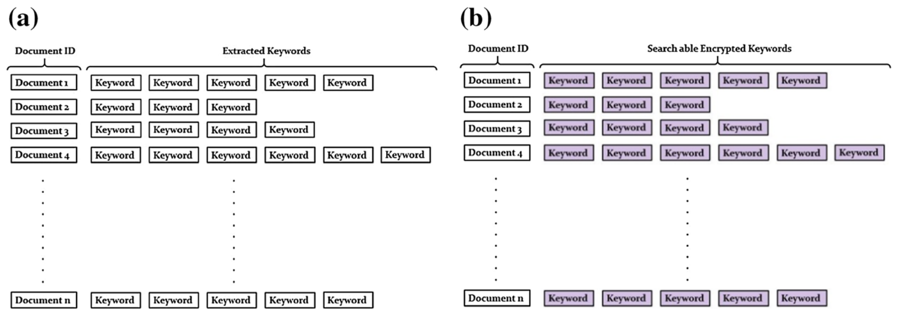


Fig. 1 Data structure of Searchable Encryption. **a** Index and **b** secure index

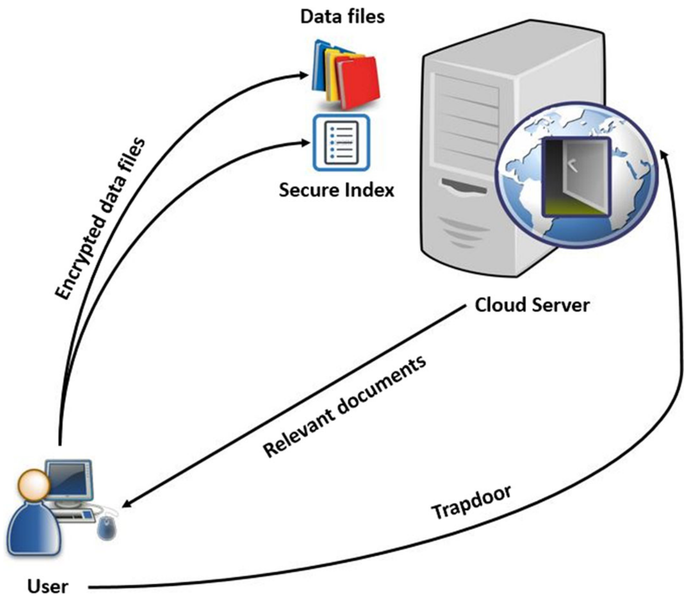


Fig. 2 Secure search on encrypted data in which data owner and data user are same

stored in cloud storage. Figure 2 depicts the overall process of secure searching from encrypted documents where data owner and data user are same person.

On the other hand, Fig. 3 illustrates the situation where data owner and the data user are different person. In this case, data owner constructs the index with the extracted keywords from the documents, encrypts the documents, and generates secure index from the index. After that, the data owner uploads the encrypted documents and the secure index to the cloud server. During searching time, the data user sends query keywords to the data owner and receives trapdoor of the query keywords from the data owner. Then the data user sends the trapdoor to the cloud server, cloud server processes the resulted documents with the trapdoor and sends the resulted documents back to the data user. This is what happens in SE in data outsourcing scenario when the data owner and the data user are not the same person. Here, the process of generating

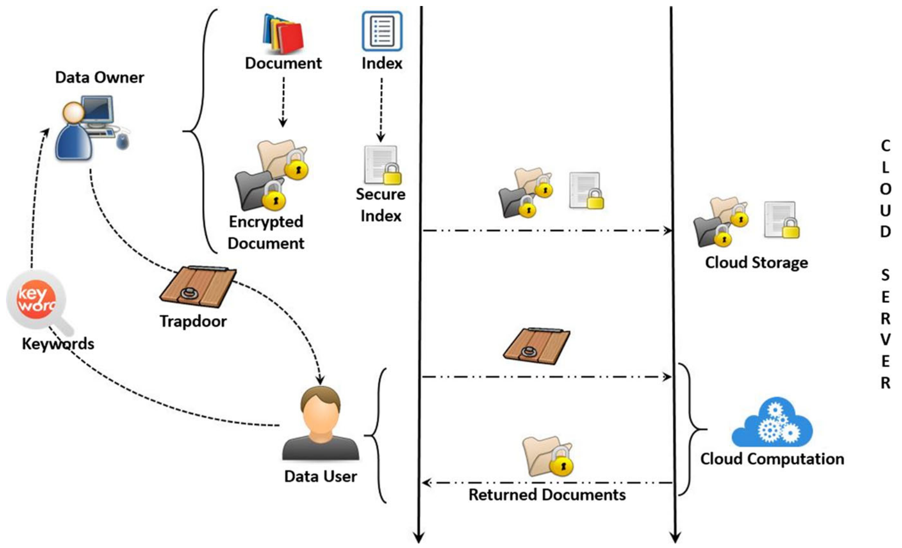


Fig. 3 Data owner encrypts documents and generates Secure Index from Index and sends it to server. Then owner himself or his authorized agent (Data User) searches encrypted document with trapdoor of query keywords

trapdoor from the query keywords by the data user with the assistance of data owner is known as search control and the process of distribution of decryption key by the data owner to the data user is known as access control. However, both search control and access control are out of scope of this work.

In any searching mechanism, effective data retrieval needs relevance ranking, instead of returning unsorted documents only for containing query keywords [15, 16]. To preserve privacy, searching mechanism should not reveal any ranking related information. Moreover, it is also necessary to support multiple keywords search, i.e., multi-keyword query. As a widely used practice (like Google or Bing search query), searcher tends to provide a set of representative keywords instead of only one to fetch the most conforming data. Each keyword in the search query helps to shrink the result further. "Coordinate matching" [17] is the widely used technique of multi-keyword ranked search and widely accepted in the plaintext information retrieval (IR) community. Nonetheless, the technique of implementing it in encrypted data remains a difficult task because of inherent security needs, including various strict requirements like the privacy of data, index, and keyword.

The most common approach in this trait is to represent each document (index) or query with a vector and to exploit secure k-nearest neighbours (kNN) algorithm for encryption of index/query vectors, at the same time, to ensure accurate relevance score calculation between index and query vectors [18–23]. Each of these works has their own security extensions and/or other additional features, however, at the core, they use a secure kNN algorithm to conceal index/query vectors.

Secure kNN algorithm has its own limitations of privacy breaches [24] and computing relevance score for each document is time-consuming as it keeps the user waiting while searching for relevant documents. The secure kNN centric schemes are susceptible to correlation attack. As they multiply a vector with a matrix which is a homomorphic transformation (i.e. ratio of distances between two pairs of points are preserved). So an adversary can compute the Cartesian distance between two vectors and if the distance is below a predefined threshold value then the adversary can conclude that the two vectors are somehow correlated. This privacy breach is termed as a correlation attack. Correlation attack is most formidable for the trapdoor. Despite randomness in trapdoor, correlation attack assists an attacker to find the same trapdoors.

On the contrary, in single keyword ranked search schemes [25–27], relevance scores between a document and a keyword is encrypted using order preserving encryption (OPE) [28, 29]. However, OPEs are not secure against chosen plaintext attack (CPA) and vulnerable to range exposure attack meaning that OPEs expose a range of a plain text within a certain probability [27].

In our observation, deterministic (multi-keyword ranked) searchable encryption schemes are susceptible to various statistical attacks [30]. These attacks may lead to a privacy breach where an attacker, instead of gathering actual data, obtains some statistical information such as term frequency, access pattern, search pattern. Search pattern, which is a sequence of trapdoor (of query keywords) and the access pattern, which is a sequence of returned document lists (in response of the trapdoors) are very crucial statistical information [31]. Search pattern or access pattern is used to predict a keyword with high probability. In turn, these information helps an adversary to infer the content of the index or query or the actual data.

To counter statistical attacks, we propose a scheme titled Countering Statistical Attack in Cloud-based Searchable Encryption (CSA-CSE) that constructs an inverted index based on a pair of keywords and generates randomized trapdoor. In CSA-CSE, the index is a collection of keyword pairs where each pair is followed by an array of document IDs in the order of relevance score of that pair of keywords. To hide the keywords in the index, CSA-CSE resorts to keyed hash function (i.e. NMAC or HMAC). During the searching time, trapdoor of query keywords is built using same hash function and same key with the addition of some randomness to ensure trapdoor indistinguishability. Additionally, the searching process should be faster as this operation takes place keeping the data user waiting for its result. CSA-CSE exemplifies a multi-keyword ranked searchable encryption scheme in symmetric key setting to cover secure data outsourcing problem in the cloud. Unlike most existing schemes, CSA-CSE strives to repel statistical attacks using unique data structure of secure index and randomized trapdoor. The contributions of this work can be summarized as following:

- We define security properties of the multi-keyword query and ranked searchable encryption in order to counter statistical data leakage from secure index, trapdoor and result.
- We propose a scheme that resists statistical information leakage from searchable encryption.
- The performance of the proposed scheme is validated through experiments on a RFC database.

The organization of this paper is as follows: Sect. 2 describes related work. System model, threat model, necessary properties for searchable encryption are elaborated in

Sect. 3. Section 4 elucidates the proposed scheme. Security analysis of the proposed scheme and experimental results are discussed in Sect. 5. We conclude the paper with some future research directions in Sect. 6.

2 Related Work

Searchable encryption enables the user to encrypt data before outsourcing to cloud server and to search on encrypted data securely. It can be broadly categorized based on cryptographic primitives into symmetric and asymmetric key settings. Mostly, data outsourcing and data sharing scenarios are suitable for SE in symmetric [32–35] and asymmetric key [14, 36–38] respectively. This focus of our research is related to SE using symmetric key cryptosystem.

Song et al. [35] described the problem of searching on encrypted data for the first time. He proposed a solution and provided proof of it. Their scheme is provably secure considering the fact that, server learns nothing about the original text given only the ciphertext. At the same time, they prevent the server from searching any arbitrary keyword and they keep the searching keyword hidden to the server. However, their algorithm takes O(n) number of stream cipher and O(n) number of block cipher operations to search a keyword, where n is the length of a document. Hence, this scheme is inefficient.

A pioneer work in [34] introduced index-based solution for searching on encrypted data. This work defined the secure index and formulated security notion based on semantics security against adaptively chosen keyword attack. Resorting to pseudo-random function and bloom filter, this scheme generates an indistinguishable index. However, the trapdoor is deterministic and observing search result, the server can deduce relationship among related documents.

Cheng et al. [32] relied on pseudo-random bit stream to mask keyword index of each file and sends it to server. Later the scheme sends server short seed to retrieve particular portion of the index while keeping other portion pseudo-random. Their scheme named Privacy Preserving Searches on Encrypted Data (or PPSED in short) uses pseudo-random permutation and pseudo-random functions to build the secure index. PPSED maintains a list of document ID versus bit array with a length of keyword size for that document. Bit array for each document holds the presence of keywords in that document, which is masked using pseudo-random bits. The user sends this list and encrypted documents to the server. Later at time of searching, the user uses short simple seeds to as a trapdoor to search a keyword.

Curtmola et al. [33] proposed a scheme that maintains a reverse index which is a list of keywords where each keyword is succeed by a document list containing that keyword. The scheme maintained an array in which each element contains the address of the next element and a decryption key for it. The master data of each keyword is preserved in a look-up table. All the data residing in the lookup table and reverse index are encrypted. Using trapdoor of a query keyword, the server searches the lookup table then finds a link between all the documents that consist of the keyword concealed in the trapdoor.

Ranking among search result of SE plays a crucial role to minimize network traffic and post-searching processing. A common approach for ranked search with single query keyword utilizes both the advancement of information retrieval and crypto community [25–27]. Wang et al. [26] used same way for ranking among search result. To get relevance (relevance score, RS) between a keyword and a document they used well known term-frequency (TF) $\times$ inverse-document-frequency (IDF) [17] and to hide the relevance score yet to provide server side ranking, they harboured on a modified version of order preserving encryption [28, 29]. However, OPE and any scheme using OPE are susceptible to range exposure attack.

All these schemes are early works in searchable encryption and show a new direction of research but lack many features in modern searchable encryption such as ranked search, multi keyword search and fuzzy search. One of the recent work [39] proposed by Fu et al. supports multiple keywords search and allows users to search with misspelled keywords or typos. However, their scheme resorts to bloom filter and suffers from false positive.

Multi-keyword ranked search is of paramount importance as it is what happens in real-world search, where user queries using multiple keywords and expects some ranked result based on relevance with query keywords. To address this problem (i.e. multi-keyword ranked search) over encrypted data, many researchers contributed in the literature [18, 21–23]. All these works used vector space model for mapping relevance score of each keyword of a particular document. To calculate relevance score between a document and a keyword, they utilized TF $\times$ IDF model [17], and to conceal these scores and yet to calculate accurate similarity (inner product or cosine metric) with query keyword, they used secure kNN computation [40]. Additionally, apart from this common calculations, each work has its own singularity in terms of secrecy, efficiency, functionality. For example, in order to achieve search efficiency, Sun et al. [22] used multi-dimensional binary tree based index structure and algorithm, aside from, Xia et al. [23] used tree-centric index and greedy depth-first search algorithm.

As noted previously, almost all multi-keyword ranked search schemes use vector space model, one of the relevance score calculation rules and secure kNN algorithm to comply multi keyword search with ranking resulting documents. Besides, each of them enhances the scheme with either efficiency or other additional features. Among them, Xia et al.'s [23] proposed a vibrant scheme referred to as Dynamic Multi keyword Ranked Search (DMRS). Their scheme is dynamic in a sense that it can handle the update of documents i.e. insertion/deletion of new/existing documents efficiently. Resorting to the vector space model and the TF $\times$ IDF model, they construct the index vector of each document and generate the query vector with query keywords. To encrypt the index and query vectors they utilize secure kNN algorithm that preserves relevance scores and ensures exact relevance score calculation between the query and index query vectors. Many modern multi-keyword ranked search schemes are almost same up to this portion. In an addition Xia et al. build a special tree-based index structure and formulates a Greedy Depth First Search algorithm that achieves sublinear search time. To prevent a statistical attack they introduced phantom terms into the index vector blinding search result. However, Greedy Depth First Search algorithm does not work perfectly in certain cases and the addition of phantom terms may result in the wrong document list. Nonetheless, secure kNN algorithm centric schemes may fall vulnerable to correlation attack.

Unlike most of these works, we present a multi keyword ranked search scheme that neither relies on OPEs nor on secure kNN algorithm with a view to resisting range exposure and correlation attacks respectively. Instead it resorts to inverted index [41] construction of secure index and incorporates randomness into trapdoor to baffle attacker. Finally, for evaluating performance of the proposed scheme, it compares the proposed scheme with secure kNN centric Multi keyword Ranked Searchable Encryption (MRSE) schemes.

3 Problem Formulation

This section formally defines the system model, threat model with attacker's capability and design goal. CSA-CSE adopts a system model of data outsourcing context and threat model signifies the attacker's domain knowledge. Apart from that, threat model also demonstrates two attacks namely, range exposure attack and correlation attack those are eminent to some existing schemes and CSA-CSE intends to repel them. Design goal of searchable encryption considers privacy at top and then efficiency for performance issue, still there exists a trade-off between privacy and efficiency.

3.1 System Model

A standard system model of searchable encryption in data outsourcing context involves three different entities: a data owner, cloud server and data user, as delineated in Fig. 4. Owner possesses a collection of documents $\mathrm{F} = \{\mathrm{f}_1, \mathrm{f}_2, \mathrm{f}_3, \ldots\ldots\ldots, \mathrm{f}_\mathrm{n}\}$ with an intention to outsource to the cloud server in encrypted format ensuring effective data utilization such as keyword based document retrieval. In our scheme, the data user who searches on encrypted data is the data owner as shown in Fig. 2. In case, data owner and data user are two different entities, there is a search control and access control mechanisms for the data user to collect the trapdoor of query keywords and decryption key to decrypt the files respectively (Fig. 4). In either cases, data owner first constructs an index I with extracted keywords from the document collection F and generates a (searchable) secure index SI from the index I. At the same time, data owner computes encrypted document collection $\mathrm{C} = \{\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3, \ldots\ldots\ldots, \mathrm{C}_\mathrm{n}\}$ from the document collection F using standard cryptosystem (e.g. RSA, AES, Salsa20). Afterwards, the data owner outsources the encrypted collection, C along with the secure index, SI to the cloud server.

Data owner authorizes the data users to access the documents. Then data user can get trapdoor of query keywords using search control mechanism. Similarly, to decrypt the encrypted documents data user gets shared secret key from data owner using access control mechanism. Both search control and access control mechanisms are out of the scope of this paper.

Cloud server preserves the encrypted document collection, C and the searchable secure index, SI. Once it receives a search request with a trapdoor TD (along with

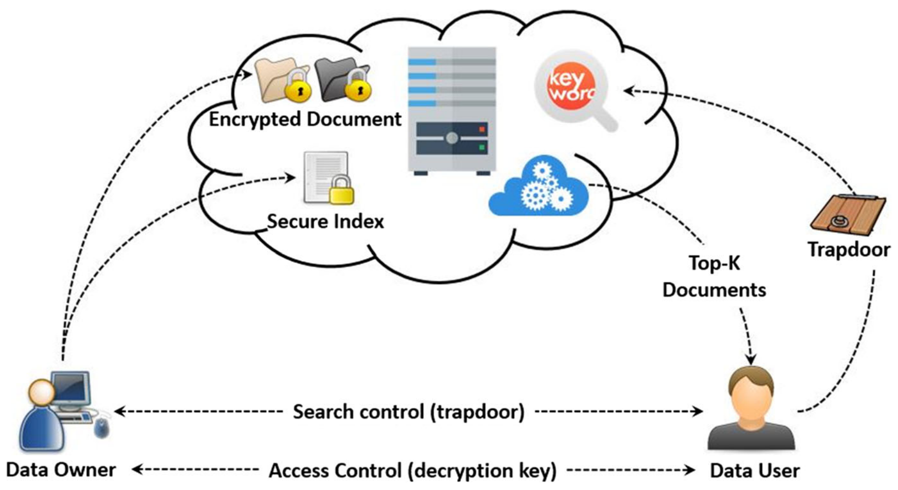


Fig. 4 System model

optional parameter, k) from data user, the cloud server performs a search over the SI, and finally returns the corresponding collection of top-k ranked encrypted documents. To protect access pattern, cloud server sends a mix of extra documents and exactly desired documents. So upon receiving returned documents, data user needs to eliminate the extra documents. Cloud server, who is responsible for storing encrypted documents and secure index and executing search operation against a trapdoor, should be furnished with a faster search operation because the search operation takes place keeping data user waiting for its result. Thus a faster search result is intended for a better user experience.

3.2 Threat Model

To design a searchable scheme, different research communities consider cloud server as honest but curious, namely, cloud server follows the Service Level Agreement properly but may pry into users' data [18, 26, 42]. Specifically, the cloud server fairly and accurately follows designated protocols. Meanwhile, it is curious to analyze the ciphertext of the encrypted documents, secure index, trapdoor and outcome of the search operation to retrieve any statistical data in order to figure out underlying information. Relying on how much information the cloud server possesses, one can classify the two threat models [18, 23], namely, known cipher text model and known background model.

3.2.1 Known Ciphertext Model

In this model, the server or other potential adversary only knows the encrypted document collection, C, the secure index, SI, and the search trapdoor, TD. All these data are in encrypted form, thus possesses a cover onto it. Apart from this, the server can

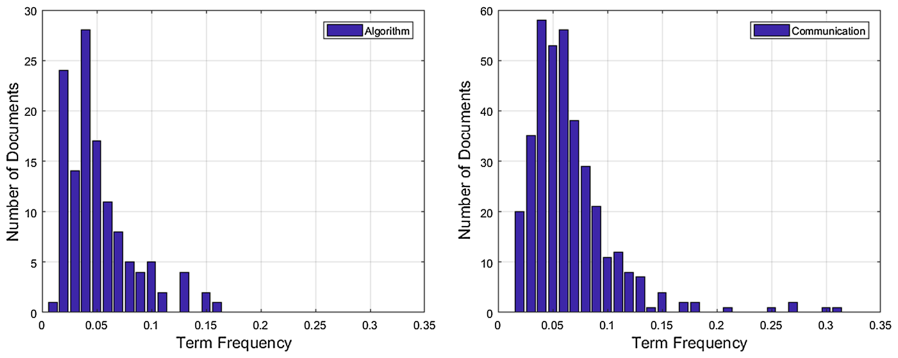


Fig. 5 Term Frequency distribution of two keywords ALGORITHM and COMMUNICATION

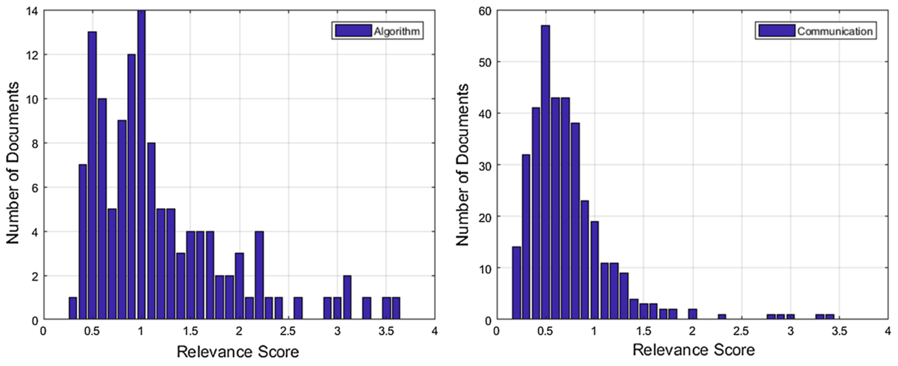


Fig. 6 Relevance dcore distribution of two keywords ALGORITHM and COMMUNICATION

compute search result between the secure index and any trapdoor. Hence, the server can launch ciphertext only attack (COA) [43] in this model.

3.2.2 Known Background Model

In contrast to known ciphertext model, the server is stronger in terms of data possession in the known background model. Besides having ciphertexts of a different component of searchable encryption, the server manages to have some additional statistical information, such as the term frequency statistics of the document collection, index/query keyword frequencies, search pattern of keywords. For example, the statistical information records how many documents are there for each term frequency of a specific keyword in the whole collection, as shown in Fig. 5, which could be used as the keyword identity [23]. Again, it reveals how many documents are there for each relevance score of a specific keyword in the whole document collection, as shown in Fig. 6. Furnished with such statistical information, the server can deduce or even identify certain keywords through analyzing prudently [26, 27, 42].

Furthermore, single keyword ranked search scheme using OPE suffers from range exposure attack and multi keyword ranked search scheme using kNN algorithm is vulnerable to correlation attack. CSA-CSE devotes to foil these attack. In this portion we explain these two attacks.

A. A Range Exposure Attack

OPE or any scheme that uses OPE is susceptible to range exposure attack. It reveals order/rank information of a plaintext that is, for two plaintext n_1 and n_2; if $n_1 < n_2$ then $OPE(n_1) < OPE(n_2)$. In a range exposure attack, an attacker determines the range of a plaintext d with certain percentage p% . For example, if an attacker knows $OPE(n_1) = C_1$ and $OPE(n_2) = C_2$ (where, $n_1 < n_2 \geq C_1 < C_2$). Then observing another ciphertext C_k between C_1 and C_2, the attacker concludes that, plaintext of C_k lays in the range $]n_1, n_2[$. Such information leakage in referred to as range exposure attack.

B. Correlation Attack

Secure k nearest neighbours (kNN) [40] algorithm splits a data vector into two random vectors and then multiplies the constituent vectors with secret matrix in order to hide the data vector. However, multiplying a vector with a matrix is a homomorphic transformation which means, ratio of distance of two vectors from a fixed point is preserved even after multiplication. Hence, to ensure data confidentiality, different schemes introduce randomness into data vector [18, 20, 23, 42]. For example, Xia et al. [23] used additional dimensions at the end of the data vector. Data vector V is having a dimension of 3 and another dimension is added making it 4. V is multiplied with secret matrix M:

$$\begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \times \begin{matrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{matrix} = \begin{matrix} v'_1 \\ v'_2 \\ v'_3 \\ v'_4 \end{matrix}$$

A single row of vector–matrix multiplication is, $v'_i = m_{i,1}.v_1 + m_{i,2}.v_2 + m_{i,3}.v_3 + \mathbf{m_{i,4}.v_4}$. Now, for more accurate vector–vector multiplication using secure kNN algorithm, the term $\mathbf{m_{i,4}.v_4}$ which is resulted from incorporated randomness must be small enough in comparison with others, $m_{i,1}.v_1 + m_{i,2}.v_2 + m_{i,3}.v_3$. Hence, observing difference between corresponding elements of two encrypted vectors V_1 and V_2 the attacker can conclude, whether two encrypted vectors are from same vector or not. This information leakage is titled as correlation attack. Correlation attack is especially formidable for trapdoors.

3.3 Design Goals

While formulating a searchable encryption, it is possible to hide information from polynomial time attacker using cryptographic means. However, unlike standard cryptosystems, multi-keyword ranked searchable encryption reveals ranked search result in a response to a multi-keyword query. On that account, an attacker can gather statistical information analyzing different component of the searchable encryption i.e. secure index, trapdoor and search result. So, one of the privacy goals of the proposed scheme is to frustrate an attacker from collecting statistical information. In order to

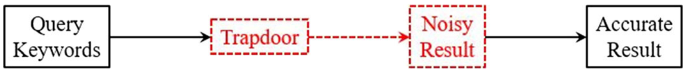


Fig. 7 Query keywords to accurate result conversion

baffle an attacker from gathering statistical information, randomness in index, trapdoor and search result can play a vital role. Randomness may result in the inaccurate result (i.e. it may return accurate documents along with some irrelevant documents), but it is a trade-off that is necessary to frustrate an attacker from accumulating non-trivial statistical information. Later, the data user can cull out the extra documents locally.

In the above system model as described in previous Sect. 3.1, the server gets the encrypted documents and secure index at the very beginning of searchable encryption lifecycle. The server can deduce correlated documents from the secure index if it is not protected properly. On the other hand, the server gets trapdoors and corresponding search results incrementally as search operation is ongoing. Figure 7 shows a search process from conversion of query keywords to acquiring accurate result documents. The data owner transforms the query keywords into trapdoor preferably by adding randomness, then trapdoor is handed over to the server. The server computes the corresponding noisy search result, after that, this noisy result is sent back to the data user. Finally, data user culls out the noise from the resulted document list. Here, trapdoor and noisy result are exposed to the server, i.e. the potential adversary. So they should contain some randomness to confuse the attacker. Observation suggests that, during trapdoor to noisy-result generation (in the server), the deterministic noise in the noisy result is better than the non-deterministic one as both trapdoor and noisy result are revealed to the adversary. Say, for a query keyword set, an accurate result is, $\mathrm{R_A} = \{\mathrm{D_1}, \mathrm{D_2}, \mathrm{D_3}\}$.

For the first run, false positive result is, $\mathrm{RF_1} = \{\mathrm{D_4}, \mathrm{D_5}\}$.

For the second run, false positive result is, $\mathrm{RF_2} = \{\mathrm{D_6}, \mathrm{D_7}\}$.

For the third run, false positive result is, $\mathrm{RF_3} = \{\mathrm{D_8}, \mathrm{D_9}\}$.

So for first, second and third runs, results are:

$$\begin{aligned} R_1 &= R_A \cup RF_1 = \{D_1, D_2, D_3, D_4, D_5\}, \\ R_2 &= R_A \cup RF_2 = \{D_1, D_2, D_3, D_6, D_7\}, \\ R_3 &= R_A \cup RF_3 = \{D_1, D_2, D_3, D_8, D_9\} \text{ respectively.} \end{aligned}$$

Now, if an attacker computes the intersection between $\mathrm{R_1}$, $\mathrm{R_2}$ and $\mathrm{R_3}$, he will obtain the accurate result or a very close approximation to the accurate result, $\mathrm{R_1} \cap \mathrm{R_2} \cap \mathrm{R_3} = \{\mathrm{D_1}, \mathrm{D_2}, \mathrm{D_3}\}$.

Consequently, CSA-CSE prefers adding a fixed set of false positive result each time for the same trapdoor.

Besides privacy, for efficiency issue, searching document list with a trapdoor needs to be faster as this search operation executes while the data users are waiting for its outcome. In addition, as the search operation takes place repeatedly, a good design of searchable encryption needs to make it faster.

To design a secure, efficient multi-keyword ranked search over encrypted cloud data under the above-mentioned system/threat models, the proposed scheme has the following design goals, Privacy-preserving and search-efficiency. By the phrase "privacy preserving" in searchable encryption, we mean secrecy of keyword information in SI and trapdoor must be protected. Simultaneously, inherently disclosed data e.g. trapdoor and/or resulted document list should be randomized.

A. A Privacy-preserving

The proposed scheme intends to prevent the cloud server from learning information about the document collection, the index, and the query except for their encrypted form. The proposed scheme intends to mask search and access patterns with randomness to confuse an attacker. At the same time, proposed scheme desired to withstand against correlation attack and range exposure attack. The specific privacy requirements are index confidentiality, trapdoor indistinguishability, randomness in exposed information. Their respective descriptions are presented below.

1. Index confidentiality: Index confidentiality prevents privacy breaches from secure index SI. SI must be irreversible and the underlying keyword information including the keyword itself, its relevance scores, term frequency, inverse document frequency should be protected from the cloud server.
2. Trapdoor indistinguishability: Trapdoor indistinguishability thwarts an adversary from getting underlying keyword information from a trapdoor. Like SI, trapdoor should be irreversible. And the server should not be allowed to learn whether two trapdoors are generated from same query keywords.
3. Randomness in exposed information: Due to nature of multi-keyword ranked search, server learns trapdoor and corresponding ranked result (ordered document list). There is no way to stop this information from being exposed. To frustrate server by gathering statistical information (i.e. search pattern and access pattern), trapdoor and its matching document list must contain randomness (even they are in encrypted form).

B. Search efficiency

The proposed scheme intends to achieve efficiency especially in a searching time as it executes repeatedly keeping data user waiting for its result.

4 Countering Statistical Attacks in Cloud-Based Searchable Encryption (CSA-CSE)

This section presents a scheme of multi-keyword ranked search referred to as CSA-CSE that confuses an attacker to collect various statistical information such as, term frequency, search pattern, access pattern and so on. For computing relevance score between a keyword and a document CSA-CSE resorts on the widely used TF $\times$ IDF model [17]. At the core of CSA-CSE's security, there is a special construction of two keywords termed as secure seed (SS), which is one-way transformation of the keywords. Due to inherited privacy breaches of order-preserving functions (OPF) [28,

Table 1 List of Notations

$F = \{f_1, f_2, f_3, \ldots\ldots\ldots, f_n\}$	Set of all documents
$C = \{C_1, C_2, C_3, \ldots\ldots\ldots C_n\}$	Encrypted form of F
n	Total number of documents
$W = \{W_1, W_2, W_3, \ldots\ldots\ldots W_m\}$	Set of all keywords extracted from all documents
m	Total number of keywords
k_E	Encryption key required to encrypt all the documents
k_H	Hash key
$message_1 \parallel message_2$	Concatenation of two messages, $message_1$ and $message_2$
$Hash(k_H, \cdot)$	Keyed hash [45] function that returns hash digest of input message using k_H as hash key
$f_{SS}(k_H, W_i, W_j)$	Function to compute secure seed of two keywords W_i and W_j are using a hash key k_H It takes two keywords W_i, W_j and a hash key k_H. It returns keyed hash digest of "$W_i \parallel W_j$" if $W_i \leq W_j$ or returns keyed hash digest of "$W_j \parallel W_i$" otherwise. $f_{SS}(k_H, W_i, W_j) = \begin{cases} Hash(k_H, W_i \Vert W_j);\ \text{if } W_i \leq W_j \\ Hash(k_H, W_j \Vert W_i);\ \text{otherwise} \end{cases}$
W_q	List of query keywords, it is a subset of W
TD	Trapdoor of W_q
k	Optional parameter that represents number documents to be returned as a response to a trapdoor

29, 44] or secure kNN algorithm [24, 40], CSA-CSE sidesteps them and keeps the documents in an order of relevance scores with an SS. Table 1 describes the notations used in this paper and Table 2 presents the main processes of typical searchable encryption along with their corresponding steps. Then we describe the algorithm of secure seed. The following subsections describe the index structure of CSA-CSE and the proposed CSA-CSE scheme followed by its security analysis.

```
Algorithm: Secure Seed aka. SS
Input: hash key $k_H$, two keywords $W_i$ and $W_j$
Output: secure seed of two keywords $W_i$ and $W_j$
Procedure:
If $W_i \leq W_j$ then return $Hash(k_H, W_i||W_j)$.
Else return $Hash(k_H, W_j||W_i)$.
```

Table 2 Main processes of a searchable encryption with their steps

Process	Steps
Documents → index	Extract keywords
	Calculate relevance scores
	Select keyword pairs
	Sort documents based on relevance scores
Index → secure index	Replace each keyword pair with its secure seed
	Remove relevance score from index
Query keywords → trapdoor	Mix extra (random) keyword(s)
	Select keyword pairs (randomly)
	Replace each keyword pair with its secure seed
Search	Find secure seeds (of trapdoor) from secure index
	Fetch their documents from top

4.1 Index Structure

Index in CSA-CSE is a list of keyword pairs where each keyword pair is followed by a document IDs in order of their relevance scores with that keyword pair. Unlike single keyword ranked search schemes [26, 27], CSA-CSE does not use order-preserving encryption to hide relevance score between document and keywords. Instead, it ranks the documents on the basis of their relevance score. Again, CSA-CSE considers relevance score of a document against a pair of the keyword, instead of a single keyword. It helps to get better ranking of documents and to conceal individual keyword's relevance score.

For construction of index, CSA-CSE extracts keyword set, $W = \{W_1, W_2, W_3, \ldots\ldots\ldots, W_m\}$ from all the documents. Then for each document, it computes relevance score with each keyword using TF × IDF model [17]. After that, it computes relevance score of each document with each pair of keywords. If the total number of keywords is m then there are $C_2^m = \frac{m(m-1)}{2}$ number of keyword pairs. Relevance score of a document with a keyword pairs is simply the sum of relevance scores of the document with each of the keyword in that pair. Finally, document list against a keyword pair are sorted based on their relevance with that keyword pair. In secure searchable index (SI), each keyword pair is replaced with its secure seed as computed by above $f_{SS}(k_H, W_i, W_j)$ function. Hence, SI preserves the secure seeds in place of keyword pair and IDs of ordered documents instead of (order preserving) encrypted form of the relevance scores.

As an illustration, we present a hypothetical scenario of four documents, f_1, f_2, f_3, f_4; five keyword, A, B, C, D and E; so the keyword pairs are (A, B), (A, C), (A, D), (A, E), (B, C), (B, D), (B, E), (C, D), (C, E), (D, E). To demonstrate the work-flow of secure index construction, Table 3 shows the imaginary relevance scores between documents and keywords. These numeric values of relevance scores are required for ranking of documents on the basis of relevance scores in this demonstration. Relevance score of a document with a keyword pair is the sum the of the document's relevance scores

Table 3 Relevance score of document versus keyword

Document	A	B	C	D	E
f_1	.2	.3	.25	.15	.35
f_2	.13		.27	.23	.03
f_3	.56	.09	.16	.4	
f_4		.36		.39	.43

Table 4 Relevance score between document and keyword-pair

A, B	A, C	A, D	A, E	B, C	B, D	B, E	C, D	C, E	D, E
$f_1 = .5$	$f_1 = .45$	$f_1 = .35$	$f_1 = .55$	$f_1 = .55$	$f_1 = .45$	$f_1 = .65$	$f_1 = .4$	$f_1 = .6$	$f_1 = .5$
$f_2 = .13$	$f_2 = .4$	$f_2 = .36$	$f_2 = .16$	$f_2 = .27$	$f_2 = .23$	$f_2 = .03$	$f_2 = .5$	$f_2 = .3$	$f_2 = .26$
$f_3 = .65$	$f_3 = .72$	$f_3 = .96$	$f_3 = .56$	$f_3 = .25$	$f_3 = .49$	$f_3 = .09$	$f_3 = .56$	$f_3 = .16$	$f_3 = .4$
$f_4 = .36$	$f_4 = .36$	$f_4 = .39$	$f_4 = .43$	$f_4 = .36$	$f_4 = .75$	$f_4 = .79$	$f_4 = .39$	$f_4 = .43$	$f_4 = .82$

Table 5 Ranking of documents for each keyword-pair

A||B	A||C	A||D	A||E	B||C	B||D	B||E	C||D	C||E	D||E
f_3	f_3	f_3	f_3	f_1	f_4	f_4	f_3	f_1	f_4
f_1	f_1	f_4	f_1	f_4	f_3	f_1	f_2	f_4	f_1
f_4	f_2	f_2	f_4	f_2	f_1	f_3	f_1	f_2	f_3
f_2	f_4	f_1	f_2	f_3	f_2	f_2	f_4	f_3	f_2

with individual keywords. For example, relevance score of f_3 with keyword pair (A, D) (as showed in third column and fourth row of Table 4: 0.96) is the sum of relevance score of f_3 with A (0.56) and relevance score of f_3 with D (0.4) each from Table 3. Table 4 shows the relevance scores between the documents and the keyword pairs. Finally, Table 5 shows the final step of index, where documents are sorted depending on their relevance scores with the keyword pairs. To transform the index into secure index, keyword pairs are replaced with their secure seeds.

4.2 Countering Statistical Attack in Cloud Based Searchable Encryption

In conformity with other searchable encryption schemes [11], CSA-CSE consists of four algorithms: Setup, SecureIndex, Trapdoor and Search.

SK ← Setup (λ) This algorithm takes system parameter λ and generates the secret key (SK) which is comprised of an encryption key k_E and a hash key k_H. The encryption key k_E is used to encrypt the documents using any of the standard crypto system such as symmetric (e.g., AES, 3DES, Salsa20) or asymmetric (e.g., RSA, ECC) and the hash key k_H is used to generate the secure seed of keyword pair.

SI ← SecureIndex (F, SK) This algorithm takes document list F and secret key SK and returns secure index SI. Before generation of SI, data owner extract keyword list $W = \{W_1, W_2, W_3, \ldots\ldots\ldots, W_m\}$ from F, computes relevance scores between document and keyword and generates an index similar to that described in Sect. 4.1. Finally, the data owner computes secure seeds of the pairs of keywords. Accordingly, SI is generated which is a list of secure seeds (of keyword pairs) and each secure seed is followed by a set of ranked document IDs.

TD ← Trapdoor (SK, W_q) Trapdoor is a randomized algorithm that takes query keywords W_q and secret key SK as input and returns randomized trapdoor for the query keywords. First of all, it chooses n_e numbers of extra keywords ($n_e \in N$). The value of n_e depends on the policy of how much noise the data owner intends to add with query keywords. Then all the query keywords (including the extra keywords) are arranged linearly in a random order and a pair of keywords are formed with each adjacent keyword. First and last keywords form another pair. So each keyword contributes to two pairs. Lastly, set of secure seeds of all the keyword pairs is the (randomized) trapdoor of the query keywords W_q.

For example, if query keyword set, $W_q = \{q_1, q_2, q_3\}$, randomly chosen extra keyword is q_4 and keywords are placed in q_1, q_2, q_3, q_4 order then, randomized trapdoor TD for is W_q,

$$TD = \{f_{SS}(k_H, q_1, q_2), f_{SS}(k_H, q_2, q_3), f_{SS}(k_H, q_3, q_4), f_{SS}(k_H, q_4, q_1)\}$$

Document-IDs ← Search (SI, TD, k) Search algorithm takes the secure index, a trapdoor (TD) and optional parameter k as input and returns set of k documents that match with TD. Due to simplicity of construction of the secure index and the trapdoor, this algorithm only selects the secure seeds from SI those exist on the TD and returns k/n_{TD} number of document IDs from each list, where n_{TD} is the number of secure seed in the trapdoor. The value of k is chosen carefully to minimize the number of extra documents.

According to system model, the data owner extracts keywords $W = \{W_1, W_2, W_3, \ldots\ldots\ldots, W_m\}$ from document collection F and builds index. At the same time, the owner generates a vector of relevance scores for each document with the relevance scores of each keyword in the same order. Then the owner computes encrypted documents ($C = \{C_1, C_2, C_3, \ldots\ldots\ldots, C_n\}$) appending the corresponding relevance score vector with each document. Finally, he generates secure index from the index and outsources the encrypted document collection (C) and the secure index (SI) to the cloud server. During searching time, server incorporates secure index with trapdoor, TD (sent by the data user) to find the intended document IDs and sends the relevant encrypted documents back to the data user. The user decrypts the returned documents and finds the exact documents using the relevance score inside the documents locally. Like other searchable encryption schemes, in CSA-CSE, there is an inverse relationship between secrecy and efficiency.

4.3 Security Analysis

This section analyses the security of CSA-CSE according to three privacy requirements defined in Sect. 3.3: index confidentiality, trapdoor indistinguishability, randomness in exposed information.

1. Index confidentiality: In CSA-CSE, the index contains a list of secure seed (of keyword pair) followed by a list of document IDs in order of their relevance. One-way hash function protects the privacy of keyword pair inside the secure seed. The ranking of documents is naturally exposed over a long period of search operations. Hence, exposing the rank or order information of documents in the secure index does not reveal any protected information.

As of relevance score, CSA-CSE does not keep these values not even in order-preserving-encrypted form in the secure index. Instead, it ranks the documents according to relevance scores. So, unlike single keyword ranked search schemes [26, 27] those use order-preserving encryption to hide relevance scores, CSA-CSE does not suffer from range exposure attack. On the other hand, unlike some multi-keyword ranked search schemes [18, 21–23], CSA-CSE does not contain encrypted form (with secure kNN algorithm) of relevance score vectors. So the index of CSA-CSE is not vulnerable to correlation attack.

To facilitate ranking documents, CSA-CSE keeps document IDs in the order of their relevance scores with keyword pair. Despite every possible step, if an adversary tries to guess keyword pair W_i, W_j and relevance score RS with a document f_k, still it might not be possible to find the exact relevance scores of two keywords W_i, W_j with the document f_k (RS_{W_i} and RS_{W_j} respectively), because there are millions of options to make $RS = RS_{W_i} + RS_{W_j}$.

2. Trapdoor indistinguishability: In CSA-CSE, trapdoor generation process is a randomized algorithm. For a query of m_q number of keywords, if number of noisy keywords is m_n and total number of keywords is m then there are $\left(m_q + m_n - 1\right) \times \left(m - m_q\right)$ different options for the query (given, $m_q + m_n \geq 4$). So for same query keyword set, CSA-CSE generates different trapdoors in different times. Similar to secure seed of index, trapdoor also protects it underlying keywords using one-way hash function. At the same time, selection of noisy keyword(s) and combination of keywords are performed at random which makes the adversary clueless about origination query keywords.
3. Randomness in exposed information: In CSA-CSE, it is possible for a trapdoor of a query keyword set to be valid trapdoor for another query keyword set which helps to obfuscate search pattern of searchable encryption. For example, if a query has W_1 and W_2 query keywords and keyword W_3 is added as a noise then $W_q = \{W_1, W_2, W_3\}$. Here, trapdoor TD for W_q is, $\{f_{SS}(k_H, W_1, W_2), f_{SS}(k_H, W_2, W_3), f_{SS}(k_H, W_3, W_1)\}$. Similarly, TD will be trapdoor of query keywords W_2, W_3 and if W_1 is chosen as noisy keyword. Thus same probabilistic trapdoor produces same results (i.e. document IDs) which, in turn, obfuscates the accurate access pattern. In consequence, CSA-CSE protects search pattern and access pattern even though they are supposed to get revealed characteristically. However, this protection comes by the price of communication overhead and post processing computation. Instead of getting accurate results (i.e. encrypted documents), the data user gets some undesired documents along with expected documents. Then the data user can find and remove the extra documents locally.

5 Experiment Results and Analysis

In this section, we analyze the performance of the CSA-CSE by implementing it using Java language on a Windows10 machine with a Core2 CPU running at 3.33 GHz. We used Request for Comments database (RFC) [46] similar to Fu et al.'s scheme [39]. We downloaded more than 8000 files from RFC database and extracted more the 500 keywords from those files. We compare the performance of CSA-CSE with the core of other multi-keyword ranked search (MRSE) schemes those use vector space model, TF $\times$ IDF model and secure kNN algorithm [18–21, 23, 26, 39, 42]. Performance is compared in terms of execution time and memory space required to store data.

5.1 Efficiency

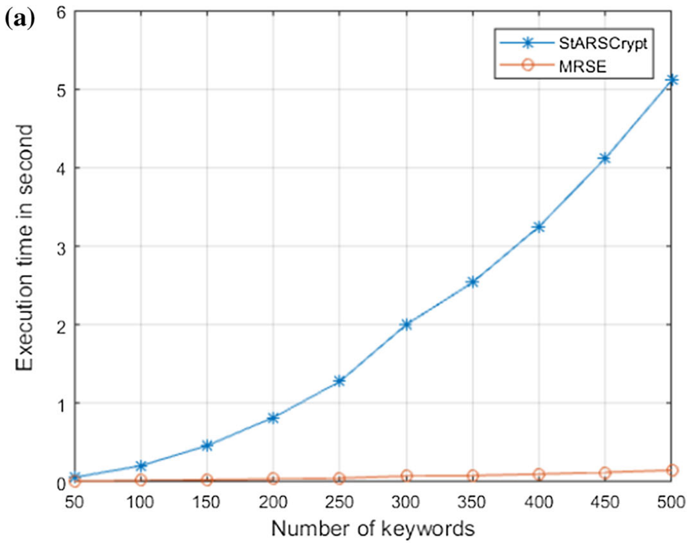
Since, both CSA-CSE and other MRSE schemes have four different algorithms. Each of the algorithms has different purposes in the scheme: key generation, secure index construction, trapdoor generation and search operation. This research compares CSA-CSE with other MRSE considering last three algorithms. It provides asymptotic analysis as well as execution time comparison on test data.

A. A Index Construction

In CSA-CSE, the index takes the form of an inverted index where index is a list of keyword pairs with each keyword pair followed by the document collection in order with relevance scores. Thus, to construct index, it requires choosing two keywords out of m keywords: there are $C_2^m = \frac{m(m-1)}{2}$ number of keyword pairs in total. For calculating relevance score of a document with each keyword pair, it needs to add the relevance scores of the two keywords with that document. Hence, for $\frac{m(m-1)}{2}$ number of keyword pairs, it requires $\frac{m(m-1)}{2} \times n$ number of addition operations, given, the total number of the document is n. Then, for each keyword pair, documents collection is sorted based on their relevance score. It necessitates total $\frac{m(m-1)}{2} \times n \log_2 n$ number of comparison operations. Finally, to turn the index into secure index, CSA-CSE computes secure seeds of the keyword pairs which requires $\frac{m(m-1)}{2}$ times of execution of $f_{SS}(k_H, \cdot, \cdot)$ function. As a whole, the time complexity for index tree construction is $O\left(m^2 n \log_2 n\right)$.

Besides, at the core of MRSE schemes' secure index construction, there exists vector space model for index representation and secure kNN computation to encrypt the index vector. To construct a secure index of a document, it computes multiplication between a vector and a matrix two times. So for n number of documents, m length vector and $m \times m$ size matrix, the time complexity of secure index construction is $O\left(m^2 n\right)$. Apart from this, each of MRSE schemes has their own computational overhead of additional security or efficiency purpose. This research compares CSA-CSE's secure index construction with the core part of MRSE schemes' secure index construction, keeping MRSE schemes at advantageous position. Because, different MRSE schemes have their own functionalities causing extra computation cost [21–23, 42].

Approximately, the execution time for constructing secure index depends on the total number of documents F and the total number of keywords in dictionary, W. Figure 8a, b show that the time cost of index tree keeping total document number (1000)



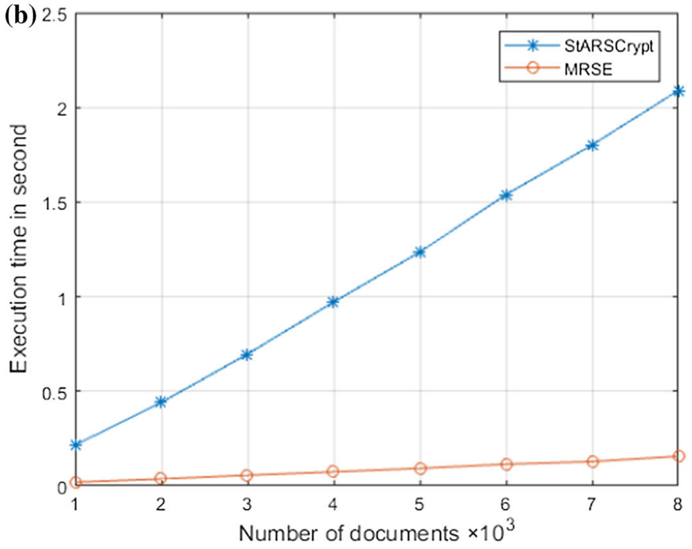


Fig. 8 **a** Secure index construction time comparison for fixed number of documents (1000 documents). **b** Secure index construction time comparison for fixed number of keywords (100 keywords)

and total keyword number (100) constant respectively. The two figures demonstrate that, CSA-CSE's index generation time curve more sensitive to the increase of keyword number than that of document number. It compiles the time complexity expression of index construction. In both figures, MRSE schemes have upper hand over CSA-CSE. CSA-CSE takes more time for building secure index because it sorts the entire document list many times, more precisely $\frac{m(m-1)}{2}$ times. Despite long generation time of secure index, CSA-CSE adopts this technique because it is one-time operation.

Furthermore, for storing secure index into cloud server, CSA-CSE requires $\frac{m(m-1)}{2} \times n$ times the size of a document ID. That is, if size of a document ID is one byte (8 bits) then it requires $\frac{m(m-1)}{2} \times n$ bytes to preserve the entire secure index in the server. Again, other MRSE schemes need only mn times the size of a document ID, keeping MRSE clearly ahead of CSA-CSE. However, $\frac{m(m-1)}{2} \times n$ is supposed to be a smaller amount in comparison with the size of actual encrypted data and has minimal storage overhead in the cloud server. CSA-CSE trades off computation/storage overhead of secure index in exchange of privacy and efficiency.

B. Trapdoor Generation

The generation of trapdoor from query keywords involves selecting extra query keyword(s), arranging all query keywords in an arbitrary order, constructing keyword pairs taking two consecutive keywords and another pair taking first and last keywords and finally, computing secure seed for each of keyword pairs. These jobs can be performed instantly. The only operation that needs computation is computing secure seeds of the keyword pairs. For instance, if there are three query keywords and an extra keyword is chosen to randomize the trapdoor then it needs to compute secure seed function four times. That is equal to a total number of keywords in the query. On the other hand, typical MRSE scheme split the vector into two vectors and then for each vector, multiply a vector of size m with a matrix of size $m \times m$ causing the complexity of trapdoor generation to $O(m^2)$. This technique is taken from secure kNN algorithm [24]. Figure 9 illustrates the trapdoor generation time comparison between CSA-CSE and typical MRSE scheme, where trapdoor generation time grows polynomially with the number of keywords, that of CSA-CSE remains almost constant. Different MRSE schemes adds addition operations into the core operation of trapdoor generation leaving them with more computation cost.

C. Search Operation

Search operation in CSA-CSE requires only to select intended documents from a pool of document collection sorted by relevance scores with different keyword pairs. As this operation involves no computation, it executes momentarily making it faster than its peers. In contrast, at the core of MRSE scheme's search operation involves two vector multiplications for all the documents. Some schemes such as Xia et al.'s [23] scheme adopts index tree construction and corresponding greedy depth-first search algorithm that enables sublinear search operation. However, for some special case, their technique costs more than linear time. Figure 10 displays the searching time comparison between CSA-CSE and MRSE schemes. With growing number of documents, searching time increase, while CSA-CSE's searching time depends only on the number of keywords (m) in the dictionary.

5.2 Search Accuracy

Due to inherent nature of searchable encryption with multi-keyword ranked search, it has to reveal encrypted documents ranked with a trapdoor. Thus there is no way to hide search pattern or access pattern in an accurate searchable encryption scheme [47]. Consistency has an inverse relation with privacy in searchable encryption. Nonetheless,

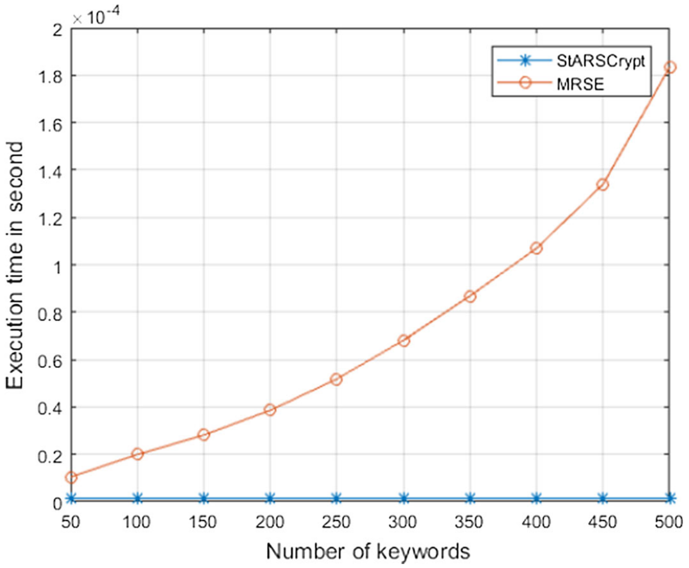


Fig. 9 Trapdoor generation time comparison

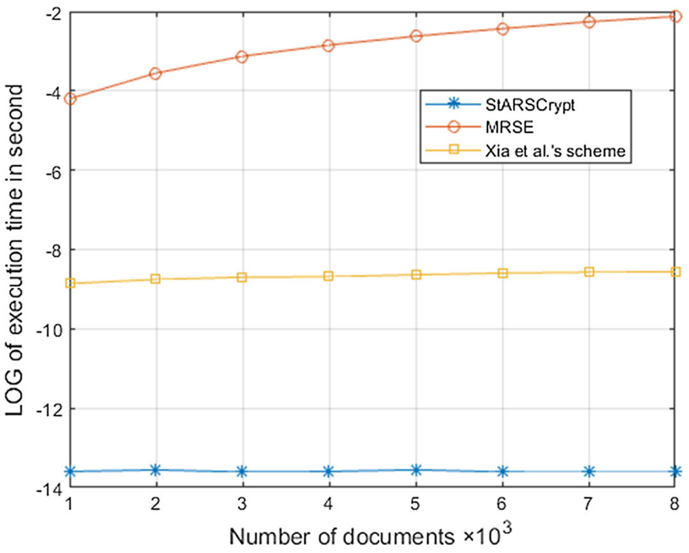


Fig. 10 LOG of execution time for a dictionary with 100 keywords

noisy trapdoor and noisy document list originating from it can vanquish an adversary to collect such statistical information. Many existing schemes add noise to the trapdoor and/or resulted in document list in order to hide the statistical information [14]. Similarly, CSA-CSE's approach incorporates noise to both trapdoor and its result. Cost emanated from the noise is a trade-off in exchange for privacy. To measure the number of extra documents that come with the result of a trapdoor, authors con-

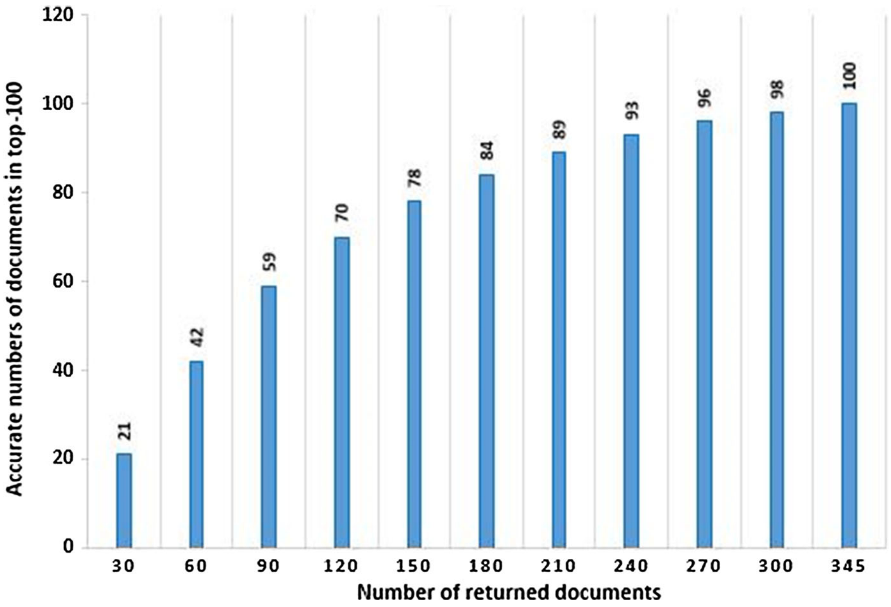


Fig. 11 Number of returned documents and exact number of top-100 documents for query keywords set {"algorithm", "communication", "protocol"}

ducted an extensive experiment on the RFC database [46]. For illustration, consider three query keywords "algorithm", "communication" and "protocol". Figure 11 shows actual number of top-100 documents against different number of returned documents, considering equal weight for each of the keywords i.e., if a user wants top-100 documents with these three query keywords, and query top-90 documents for each of the query keywords (making 90 + 90 + 90 equals 270), the user will get 96 documents those belong to exact top-100 documents. So rest 174 documents are extra documents. On the other hand, to get top-100 documents, the system has to return 315 documents, top-115 documents by each keyword. Thus, returns 245% extra documents, however, it is a tread off for privacy. Once all the encrypted documents come to the user, user decrypts and eliminate the unwanted data locally. For this purpose, each encrypted document contains a vector of relevance score for all the keywords in the dictionary.

6 Conclusion and Future Research Direction

In this paper, we have proposed a secure and efficiently computable multi-keyword ranked searchable encryption scheme named CSA-CSE with an aim to protect the privacy of statistical information. Instead of using an encryption technique (i.e. order-preserving encryption), CSA-CSE resorts on sorted order of the documents based on relevance score. Again, instead of ranking documents based on relevance score with a keyword, CSA-CSE uses relevance score with a pair of keywords. Thus original relevance score remains beyond the trace of the cloud server, the potential attacker. Though CSA-CSE conceals the keywords and relevance scores in secure index, it increases the secure index construction time. However, construction of secure index is one-time job and this extra time can easily be ignored. At the same time, this approach makes the trapdoor generation and searching operations extremely fast, which is desired in searchable encryption as the two operations take place keeping the user waiting for its result.

Search pattern and access pattern in searchable encryption get exposed by definition and to the best of authors' knowledge, the only way to confuse an attacker about the exposure is randomization. To protect the search pattern and access pattern, CSA-CSE randomizes the trapdoor and the resulted documents. To protect these patterns and to foil range exposure and correlation attacks, CSA-CSE does not utilize neither OPE nor kNN algorithms. Instead it resorts to keyword-pair centric inverted index ordered by relevance scores. Thus the privacy of search pattern and access pattern comes with the cost of transmission and post-processing delay. However, it is necessary to confuse an attacker accumulating search statistics. This extra load can be engineered by sending the encrypted relevance scores first, then sending the encrypted documents on request. A more robust searchable encryption scheme needs controlled and/or dynamic noise into trapdoor and resulted document list.

Still, there are a lot of challenges of multi-keyword ranked searchable encryption. Such scheme requires being dynamic to add/delete documents as the owner wants. This scheme need to be adapted for multi-user setting, i.e. multiple users' ability to search. Here, user revocation is a major issue to prevent an unsubscribed user from accessing further. To simulate a practical scenario, the robust cryptographic approach is necessary.

Acknowledgements This research work was partially supported by the Faculty of Computer Science and Information Technology, University of Malaya under a special allocation of the Post Graduate Fund for RP036 (A, B, C)-15AET project. This work is also supported by the Deanship of Scientific Research at King Saud University through Research Group number RG-1435-051.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., et al.: A view of cloud computing. Commun. ACM **53**, 50–58 (2010)
2. Radu, L.-D.: Green cloud computing: a literature survey. Symmetry **9**, 295 (2017)
3. Takabi, H., Joshi, J.B., Ahn, G.-J.: Security and privacy challenges in cloud computing environments. IEEE Secur. Priv. **8**, 24–31 (2010)
4. Zhou, Y., Zhang, D., Xiong, N.: Post-cloud computing paradigms: a survey and comparison. Tsinghua Sci. Technol. **22**, 714–732 (2017)
5. Yaqoob, I., Ahmed, E., Ur Rehman, M.H., Ahmed, A.I.A., Al-garadi, M.A., Imran, M., et al.: The rise of ransomware and emerging security challenges in the Internet of Things. Comput. Netw. **129**, 444–458 (2017)
6. Feng, D.-G., Zhang, M., Zhang, Y., Xu, Z.: Study on cloud computing security. J. Softw. **22**, 71–83 (2011)
7. Kamara, S., Lauter K.: Cryptographic cloud storage. In: International Conference on Financial Cryptography and Data Security, pp. 136–149 (2010)
8. Ali, M., Khan, S.U., Vasilakos, A.V.: Security in cloud computing: opportunities and challenges. Inf. Sci. **305**, 357–383 (2015)
9. Fernandes, D.A., Soares, L.F., Gomes, J.V., Freire, M.M., Inácio, P.R.: Security issues in cloud environments: a survey. Int. J. Inf. Secur. **13**, 113–170 (2014)

10. Wang, T., Zhou, J., Chen, X., Wang, G., Liu, A., Liu, Y.: A Three-layer privacy preserving cloud storage scheme based on computational intelligence in fog computing. IEEE Trans. Emerg. Top. Comput. Intell. **2**, 3–12 (2018)
11. Han, F., Qin, J., Hu, J.: Secure searches in the cloud: a survey. Future Gener. Comput. Syst. **62**, 66–75 (2016)
12. Au, M.H., Liang, K., Liu, J.K., Lu, R., Ning, J.: Privacy-preserving personal data operation on mobile cloud—chances and challenges over advanced persistent threat. Future Gener. Comput. Syst. **79**, 337–349 (2018)
13. Fu, Z., Ren, K., Shu, J., Sun, X., Huang, F.: Enabling personalized search over encrypted outsourced data with efficiency improvement. IEEE Trans. Parallel Distrib. Syst. **27**, 2546–2559 (2016)
14. Xu, P., Jin, H., Wu, Q., Wang, W.: Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack. IEEE Trans. Comput. **62**, 2266–2277 (2013)
15. Singhal, A.: Modern information retrieval: a brief overview. IEEE Data Eng. Bull. **24**, 35–43 (2001)
16. Berger, A., Lafferty, J.: Information retrieval as statistical translation. In: ACM SIGIR Forum, pp. 219–226 (2017)
17. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufmann, Burlington (1999)
18. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **25**, 222–233 (2014)
19. Fu, Z., Sun, X., Linge, N., Zhou, L.: Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query. IEEE Trans. Consum. Electron. **60**, 164–172 (2014)
20. Jiang, X., Yu, J., Yan, J., Hao, R.: Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data. Inf. Sci. **403**, 22–41 (2017)
21. Li, H., Liu, D., Dai, Y., Luan, T.H., Shen, X.S.: Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage. IEEE Trans. Emerg. Top. Comput. **3**, 127–138 (2015)
22. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., et al.: Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Trans. Parallel Distrib. Syst. **25**, 3025–3035 (2014)
23. Xia, Z., Wang, X., Sun, X., Wang, Q.: A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **27**, 340–352 (2016)
24. Yao, B., Li, F., Xiao, X.: Secure nearest neighbor revisited. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 733–744 (2013)
25. Swaminathan, A., Mao, Y., Su, G.-M., Gou, H., Varna, A.L., He, S., et al.: Confidentiality-preserving rank-ordered search. In: Proceedings of the 2007 ACM Workshop on Storage Security and Survivability, pp. 7–12 (2007)
26. Wang, C., Cao, N., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. Parallel Distrib. Syst. **23**, 1467–1479 (2012)
27. Zerr, S., Olmedilla, D., Nejdl, W., Siberski, W.: Zerber+r: top-k retrieval from a confidential index. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pp. 439–449 (2009)
28. Boldyreva, A., Chenette, N., Lee, Y., O'neill, A.: Order-preserving symmetric encryption. In: Eurocrypt, pp. 224–241 (2009)
29. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In: CRYPTO, pp. 578–595 (2011)
30. Wang, G., Liu, C., Dong, Y., Choo, K.-K.R., Han, P., Pan, H., et al.: Leakage models and inference attacks on searchable encryption for cyber-physical social systems. IEEE Access **6**, 21828–21839 (2018)
31. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., et al.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. J. Cryptol. **21**, 350–391 (2008)
32. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: International Conference on Applied Cryptography and Network Security, pp. 442–455 (2005)
33. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. J. Comput. Secur. **19**, 895–934 (2011)
34. Goh, E.-J: Secure indexes. In: IACR Cryptology ePrint Archive, 2003, vol. 216 (2003)

35. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, 2000. S&P 2000. Proceedings, pp. 44–55 (2000)
36. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: Computational Science and Its Applications—ICCSA 2008, pp. 1249–1259 (2008)
37. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: International Conference on the Theory and Applications of Cryptographic Techniques, pp. 506–522 (2004)
38. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Trapdoor security in a searchable public-key encryption scheme with a designated tester. J. Syst. Softw. **83**, 763–771 (2010)
39. Fu, Z., Wu, X., Guan, C., Sun, X., Ren, K.: Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Trans. Inf. Forensics Secur. **11**, 2706–2716 (2016)
40. Wong, W.K., Cheung, D.W.-l., Kao, B., Mamoulis, N.: Secure knn computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 139–152 (2009)
41. Guo, C., Chen, X., Jie, Y., Zhangjie, F., Li, M., Feng, B.: Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption. IEEE Trans. Serv. Comput. (2017). https://doi.org/10.1109/TSC.2017.2768045
42. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., 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, pp. 71–82 (2013)
43. Delfs, H., Knebl, H., Knebl, H.: Introduction to Cryptography, vol. 2. Springer, Berlin (2002)
44. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 563–574 (2004)
45. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Annual International Cryptology Conference, pp. 1–15 (1996)
46. (25/12/2017) Request for Comments. https://www.rfc-editor.org/rfc-index.html
47. Jeong, I.R., Kwon, J.O., Hong, D., Lee, D.H.: Constructing PEKS schemes secure against keyword guessing attacks is possible? Comput. Commun. **32**, 394–396 (2009)

Affiliations

M. A. Manazir Ahsan[1] · Ihsan Ali[1] · Mohd Yamani Idna Bin Idris[1] · Muhammad Imran[2] · Muhammad Shoaib[2]

M. A. Manazir Ahsan
manazir.ahsan@siswa.um.edu.my

Mohd Yamani Idna Bin Idris
yamani@um.edu.my

Muhammad Imran
cimran@ksu.edu.sa

Muhammad Shoaib
muhshoaib@ksu.edu.sa

[1] Department of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia

[2] College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia