Article

Concisely Indexed Multi-Keyword Rank Search on Encrypted Cloud Documents

Tai-Lin Chin and Wan-Ni Shih

Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106335, Taiwan; m10415018@mail.ntust.edu.tw

* Correspondence: tchin@mail.ntust.edu.tw

Abstract: With the advent of cloud computing, the low-cost and high-capacity cloud storages have attracted people to move their data from local computers to the remote facilities. People can access and share their data with others at anytime, from anywhere. However, the convenience of cloud storages also comes with new problems and challenges. This paper investigates the problem of secure document search on the cloud. Traditional search schemes use a long index for each document to facilitate keyword search in a large dataset, but long indexes can reduce the search efficiency and waste space. Another concern to prevent people from using cloud storages is the security and privacy problem. Since cloud services are usually run by third party providers, data owners desire to avoid the leakage of their confidential information, and data users desire to protect their privacy when performing search. A trivial solution is to encrypt the data before outsourcing the data to the cloud. However, the encryption could make the search difficult by plain keywords. This paper proposes a secure multi-keyword search scheme with condensed index for encrypted cloud documents. The proposed scheme resolves the issue of long document index and the problem of searching documents over encrypted data, simultaneously. Extended simulations are conducted to show the improvements in terms of time and space efficiency for cloud data search.

Citation: Chin, T.-L.; Shih, W.-N. Concisely Indexed Multi-Keyword Rank Search on Encrypted Cloud Documents. *Appl. Sci.* **2021**, *11*, 11529. <https://doi.org/10.3390/app112311529>

Academic Editor: Katarzyna Antosz

Received: 4 November 2021

Accepted: 30 November 2021

Published: 5 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: rank search; cloud; encryption; security

1. Introduction

The advances of Internet technologies have brought people the ability to access the plentiful resources on the cloud. The low-cost, powerful, and convenient services provided on the cloud have drawn people to move from their local computers to remote computing facilities. One of the most popular cloud services is cloud storage. The Cloud storage service allows people to access their data at anytime, from anywhere. It also allows data owners to share their data with authorized users. It has dramatically changed the way that people save their data and share their information.

The convenience of cloud storages also comes with great challenges and problems. The first one is the search mechanism. Since the number of documents stored in cloud storage is typically large, users usually find the documents of their interest using several keywords. Traditional keyword search methods use a long index for each document in the dataset. The index is usually denoted by a vector to characterize the presence or absence for a bunch of selected keywords. For each document, the keywords present in the document are marked as one in the corresponding entry in its index. When a user raises a query with certain keywords, the indexes of the documents are used to match for the user's request. Since a document may contain only a few keywords to characterize its contents, most of the locations in the index are zero. The long index could be a waste, and may increase the computation and communication overhead. Obviously, a concise index scheme is desired.

Another concern to prevent people from using cloud storage is the security and privacy issue, since cloud storages are usually run by a third party service provider. Data owners want to secure the risk of information leakage, and data users would like to protect their

privacy when performing search. A trivial solution is to encrypt the data before outsourcing them to the cloud, but it brings difficulty in data utilization, since the encrypted data cannot be searched like in plain text. It is also impractical to download the whole dataset and decrypt it to find the documents of the users' interest.

This paper proposes a Condensed Index Multi-keyword Search (CIMS) scheme for encrypted cloud data. Two aspects are considered for the secure keyword search problem. The first aims to reduce the size of document indexes for the multi-keyword rank search. Users can raise multiple keywords to search the documents of interest in the dataset. Since most of the terms in document indexes are zero, it is possible to condense the document indexes while preserving the information contained in the original index. Principal component analysis (PCA), which is one of the technologies to simplify data in multivariate statistical analysis, is adopted to reduce the size of document indexes. Essentially, PCA performs a linear transformation to transfer the document indexes to a new eigenspace such that the first coordinate in the new space preserves the most information about the document indexes. In contrast, the last coordinate in the new space contains the least information about the document indexes. Thus, if the later dimensions of the document indexes in the new space are omitted, only a commensurately small amount of information is lost. Therefore, by preserving only the principal components of the document indexes can effectively condense the document index and significantly reduce the index size. Actually, the scheme trades the search precision with the index size.

The second aspect is to incorporate the condensed index to a secure multi-keyword search scheme for encrypted cloud data. Both the documents and the query are encrypted. The third party cloud server actually performs the search by a pre-designed procedure in ciphertext, and cannot discover the document contents or the query information raised by users. However, the results of the pre-designed procedure are the same as that doing in plain text. The proposed CIMS scheme resolves the problem of condensing the document indexes and performing secure data search at the same time.

The contributions of the paper can be summarized in three folds.

- The sizes of document indexes are significantly reduced, with only a commensurate loss on search precision.
- A searchable encryption scheme which enables multi-keyword search is seamlessly integrated with the condensed document indexes. Data security and user privacy are protected in the popular cloud computing environment.
- A real-world dataset is collected to evaluate the performance of the proposed scheme. The results show that the proposed scheme significantly improves the search efficiency in terms of time and space.

The paper is organized as follows. Section 2 describes the related work for a secure keyword search on the cloud. Section 3 addresses the system model and illustrates the problem. The proposed scheme for condensing the document index and performing secure search on the cloud is presented in Section 4. Section 5 shows the simulation results for the performance of the proposed scheme. The paper concludes in Section 6.

2. Related Work

The issue for searching over encrypted data to protect users' privacy and data security has been studied for the recent cloud computing environment [1–7]. The goal is to prevent the untrusted cloud server to know details about the data and users' query during the search. The early work in [1] suggested that the encrypted search on the untrusted server should fulfill four features—query isolation, controlled searching, hidden query, and provably secure. Query isolation indicates that the untrusted server cannot learn anything about the data content more than the search results. Controlled searching prevents the untrusted server from searching the data without the data owner's authorization. Hidden query means that the user can ask the untrusted server to perform the search without revealing the search keyword to the server. Certainly, the scheme should be provable for secure. Conventional techniques to perform secure search over encrypted data include Searchable

Symmetric Encryption (SSE) [2,7,8] and public key-based cryptography [4,9]. Symmetric encryption is a cryptography scheme, in which the same key is used to encrypt the plaintext and decrypt the ciphertext. Searchable symmetric encryption allows the data to be stored in the third party storage, while still maintaining the capability to search for the encrypted data. For the public key-based cryptography scheme, the data content and index keywords are encrypted separately. A secret key is sent to the third party server by the data owner, and the third party server can use the key to identify whether a particular keyword is in the index, without learning anything else about the data content [4].

Goh et al. use bloom filter to construct the secure index for documents [2]. Bloom filter requires a relatively small space to store the keyword set and calculates via hash coding. Nevertheless, it also reduces the correctness of the results, since the error rate inherited the hash coding. Curtmola et al. further define the searchable encryption mechanism and proposed an index based scheme with the inverted list to link the same keywords in different files [7]. In [6], Wang et al. proposes a ranked keyword search scheme for encrypted cloud data. The returned documents are in the order of certain score function, for instance, the frequency of the keyword in the associated documents. However, these studies only support single keyword encryption search.

While single keyword search is sufficient for certain applications, multiple keyword search is more appropriate for most of the practical usage. In [10], a scheme for multi-keyword ranked search over encrypted data is presented. The scheme adopts the similarity measure based on coordinate matching, i.e., as many matches as possible, to characterize the relevance of the documents to the query. The paper also improves the proposed scheme to fulfill various privacy requirements, such as adding a small random perturbation in the query to further obfuscate the search each time. The perturbation trades off the search precision to privacy requirements. In [11], a tree-based search scheme is developed to increase the search efficiency. The above work adopts a vector based document indexes and uses inner product to measure the similarity. However, they may use a long index for a document, and reduce the space and time efficiency of the scheme.

Multi-keyword boolean search has also been studied to enrich query predicates for encryption search. Two major types of boolean keyword search are commonly studied in the literature, namely conjunctive keyword search [3,5,9] and disjunctive keyword search [12,13]. For conjunctive keyword search, the returned documents have to completely match all the query keywords. In contrast, for disjunctive keyword search, the documents that satisfy a subset of the query conditions can be used as the answer. Predicate search schemes are also developed to resolve both conjunctive and disjunctive keyword search [14,15]. Boolean search can reveal whether a keyword is present in a document. However, the relevance of a word to different documents may be different. It may not be adequate to search documents only based on the presence or absence of certain words.

Most of the schemes generate their secure indexes based on the extracted keywords from the corpus. Conventional search schemes usually perform exact matching for the document keywords with the query keywords. However, typos or the inflection of words are inevitable in the practice. Recently, much effort has been devoted to design the mechanisms for fuzzy search [16–18]. Li et al. propose a wildcard-based fuzzy search. However, each keyword is associated to $O(l^d)$ fuzzy keywords, where l is the length of the keyword and d is the computational distance. The size of the keyword index is quite large. Bing Wang et al. [18] proposes a fuzzy search scheme to deal with misspelling the query keywords by using local sensitive hash (LSH) with bloom filter. The LSH function maps similar items to the same position with high probability in a bloom filter. Thus, the search process may be able to return appropriate results associated to the corrected keywords if a typo occurs. Nevertheless, the scheme only solves one letter error mistake in the keyword. Fu et al. [19] adopt the Porter Stemming Algorithm to organize inflected words by the word stem, and tolerate formatting-inconsistent or spelling mistakes. Again, the studies do not focus on the index size, and may decrease the search efficiency.

3. Preliminary Foundation

This section defines the basic assumptions of this paper. The system model and threat model are presented. Based on the models, the investigated keyword search problem is illustrated.

3.1. System Model

The system model is comprised of three different entities, as illustrated in Figure 1; namely, data owner, data user and cloud server. The data owner has a collection of documents which will be uploaded to the cloud server. An index is generated for each document in order to efficiently search the documents by keywords which are raised later by data users. To protect the information in the documents from the peep of the cloud server, the documents and the corresponding indexes are encrypted before outsourcing to the cloud server. Finally, the data owner uploads the encrypted documents and index to the cloud server.

Figure 1. The architecture of our scheme.

When a data user wants to search the documents of his/her own interest, the data user has to ask the authorization from the data owner for the search. Based on the query keywords provided by the data user, the data owner creates the trapdoor, which is a secret key generated by encrypting the query keywords, to the data user. The data user sends the trapdoor to the cloud server where a pre-defined search procedure is executed. Since the trapdoor is also encrypted, the cloud server cannot know the search keywords and, thus, the privacy of the data user is also secured. The pre-defined procedure essentially computes the similarity between query keywords and the document indexes. The cloud server will return the corresponding encrypted documents with high similarity scores to the data user.

3.2. Threat Model

The cloud server is assumed to be “honest but curious”. It honestly provides the services, but is curious about the data and the query. In other words, it may try to reveal the content of the queries or the data stored in the cloud. In this paper, it is assumed that the cloud server is only able to access the ciphertexts of the documents, and the encrypted searchable indexes, which are stored on the server from the data owner. It is noted that the search control and access control are not within the scope of this paper. The search control supervises the authorization for users to perform document search. The access control inspects the access for the outsourced documents. In other words, only authorized users who possess a trapdoor can request searching for the documents.

3.3. Keyword Search Problem

Keyword search is a typical way for users to find data stored in the cloud. Consider that there is a collection of n documents. Let f_j denote document j and K_j denote the set of

feature words extracted from f_j . Thus, K_j can be used as an abstract of the document f_j to characterize the document. K_j is comprised of certain selected keywords which have high relationship to the content of f_j or include all the words in f_j , except those stop words like 'a', 'is', 'of', and 'that', etc. A dictionary is composed of all the featured words. Let W denote the dictionary, i.e., $W = \bigcup_{j=1}^n K_j$ and assume that W is an ordered set. Further assume that there are m words in the dictionary W . To facilitate the search, an index of length m is generated for each document. Specifically, let $\mathbf{a}_j = [a_{1,j} a_{2,j} \dots a_{m,j}]^T$ denote the index vector of document f_j , where $a_{i,j}$ is a measurement for the relation of the word $w_i \in W$ and document f_j . A commonly used measurement for the word-document relation is based on 'Term Frequency' (TF) and 'Inverse Document Frequency' (IDF), which is defined as follows:

$$a_{i,j} = \frac{n_{ij}}{n_j} \log\left(\frac{n}{d_i}\right), \quad (1)$$

where n_{ij} is the number of times that w_i present in f_j , n_j is the number of total words in document f_j , d_i is the number of documents that contains w_i , and n is the number of documents. The first term in Equation (1) is the term frequency of word w_i in document f_j . If the frequency is high, it is obvious that the word has stronger relation to the document. The later term in Equation (1) represents the inverse of the fraction of documents which contains the word w_i . It characterizes the uniqueness of a word to the corresponding documents. In general, a word, which is not present in many other documents, in a particular document is better to characterize the document. Essentially, TF-IDF characterizes the relevance of a word to a document. The higher the TF-IDF value, the stronger the relation between the word and the corresponding document.

The number of documents is supposed to be large. If a user wants to search the documents, the user can provide several keywords for the topic of interest. Assume that R is the set of the keywords raised by the user. A simple search scheme is to generate a query vector $\mathbf{q} = [q_1 q_2 \dots q_m]^T$ such that $q_i = 1$ if $w_i \in W$ is one of the raised keywords in R . Otherwise, $q_i = 0$. The query vector has the same size as the dictionary as well as the document index. To search the documents, the similarity of $\mathbf{q}$ and $\mathbf{a}_j$ for all $j = 1 \dots n$ is evaluated. The similarity can be computed by the inner product of $\mathbf{q}$ and $\mathbf{a}_j$, i.e.,

$$\mathbf{q} \cdot \mathbf{a}_j = \sum_i^m q_i \times a_{i,j}, \forall 1 \leq j \leq n. \quad (2)$$

The similarity is the sum of the TF-IDF values of the words which are raised by the user and in the document index of a certain document f_j . Since the TF-IDF indicates the relevance between the raised words and the documents, the larger the similarity, the better the corresponding documents fit the user's request. However, most of the entries in the document indexes are zero. The document index could be long, and, thus, could waste space and reduce the computation and communication efficiency.

The paper wants to achieve two goals for efficient multi-keyword search based on the above system model. The first one is to increase the search efficiency. Since the number of the documents is assumed to be large, the long document index may reduce the search efficiency and occupy more space. The second goal is to perform the search in encrypted context in the cloud, since the security of the documents and the privacy of the users should be protected from the third-party cloud server.

4. Concisely Indexed Multi-Keyword Rank Search

This section presents the solutions to achieve the two major targets of the paper. The first one is to reduce the size of the document indexes and use a short query vector to conduct the search. The second one is to perform keyword search over encrypted data in the cloud.

4.1. Index Size Reduction

It is trivial that using a small document index can increase the search efficiency, since the comparison of the query vector to the document indexes could be faster. One possible solution of reducing the size of document indexes is performing principal component analysis (PCA), which is one of the technologies to simplify data in multivariate statistical analysis, on the document indexes.

To facilitate computation, put $\mathbf{a}_j$ in a matrix A , which is shown as follows:

$$\mathbf{W} \rightarrow \begin{matrix} & \mathbf{F} \\ & \downarrow \\ \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix} \end{matrix}$$

The j th column of A is the document index of f_j , i.e., $\mathbf{a}_j$. Matrix A characterizes the relation between the words in the dictionary and the documents. Note that A is a sparse matrix in which most of the elements are zero. Suppose that each column of A is used as a data point in a high dimensional space. PCA is an orthogonal linear transformation that transforms the data points to a new coordinate system such that the projection of the data points onto the first coordinate has the largest variance, onto the second coordinate has the second largest variance, and so on so forth. Essentially, higher variance indicates that the corresponding dimension preserves more information about the original data set. In contrast, if the projection of the data points to some coordinate has low variance, less information of the data points is preserved in that dimension since the distance among the data points may be small and make it difficult to differentiate the data points. Therefore, it is possible to reduce the dimensions of a high dimensional data set to a low dimensional data set by eliminating those dimensions with low variance, such that only a commensurately small amount of information is lost.

Specifically, PCA of the matrix A can be derived as follows. Let X be the matrix which is obtained by shifting A to its column mean, i.e.,

$$X = \begin{bmatrix} \mathbf{a}_1 - \bar{\mathbf{a}} & \mathbf{a}_2 - \bar{\mathbf{a}} & \cdots & \mathbf{a}_n - \bar{\mathbf{a}} \end{bmatrix}, \quad (3)$$

where $\bar{\mathbf{a}} = \frac{1}{n} \sum_{j=1}^n \mathbf{a}_j$. PCA wants to find the axis which maximizes the variance of the data points after projecting the data points to that axis. Formally, the projection of X onto a certain axis $\mathbf{u}$ can be derived by $\mathbf{p} = (X^T \mathbf{u})$. The variance after the projection is given by

$$\frac{1}{n} \|\mathbf{p}\|^2 = \frac{1}{n} (X^T \mathbf{u})^T (X^T \mathbf{u}) = \frac{1}{n} \mathbf{u}^T X X^T \mathbf{u}, \quad (4)$$

where $\mathbf{u}$ is a unit vector. To maximize the variance after projection to a certain coordinate, the problem can be formulated as

$$\begin{aligned} \max_{\mathbf{u}} \quad & G(\mathbf{u}) = \mathbf{u}^T X X^T \mathbf{u} \\ \text{s.t.} \quad & \mathbf{u}^T \mathbf{u} = 1 \end{aligned}$$

The problem can be transformed to a nonconstraint problem using the Lagrange multiplier as follows:

$$\max_{\mathbf{u}} \quad \tilde{G}(\mathbf{u}, \lambda) = \mathbf{u}^T X X^T \mathbf{u} + \lambda (1 - \mathbf{u}^T \mathbf{u}). \quad (5)$$

Taking the gradient of Equation (5), one can have

$$\nabla_{\mathbf{u}} \tilde{G}(\mathbf{u}, \lambda) = 2XX^T\mathbf{u} - 2\lambda\mathbf{u} = 0 \quad (6)$$

$$\Rightarrow XX^T\mathbf{u} = \lambda\mathbf{u} \quad (7)$$

Thus, $\mathbf{u}$ is the eigenvector of XX^T , and the maximum $G(\mathbf{u})$ is given by

$$G(\mathbf{u}) = \mathbf{u}^T XX^T \mathbf{u} = \mathbf{u}^T \lambda \mathbf{u} = \lambda, \quad (8)$$

where λ is the corresponding eigenvalue. Consequently, $\mathbf{u}$ is the coordinate which preserves the most variance when projecting the data points onto it, and it is the first principal component. Note that projecting the data point $\mathbf{x}_j$, i.e., the j th column of X , onto $\mathbf{u}$, is equal to $\mathbf{u}^T \mathbf{x}_j$, which is the magnitude of the projection of $\mathbf{x}_j$ on the coordinate $\mathbf{u}$. Analogously, the k th principal component can be found by subtracting the first $k - 1$ principal components from X . Denote $\mathbf{u}_j$ as the eigenvector in the order of the principal components and λ_i as the corresponding eigenvalue. The resulting eigenvalues have the relation $\lambda_1 > \lambda_2 > \dots > \lambda_m$.

It is noted that for square matrix, the eigenvectors of the matrix are orthogonal. Since XX^T is square, the eigenvectors form the basis for a new space. Consequently, PCA transforms the original data set to the new space. One can write the projection in matrix form as follows:

$$T = U^T X, \quad (9)$$

where $U = [\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_m]$ and T contain the coordinates of the data points in the new space. As mentioned before, since the former principal components preserve more information of the data, one can eliminate some later principal components with only a small amount of information loss. Suppose that only the first r principal components are preserved. One can project a high dimensional data set to a lower dimensional space, i.e.,

$$T_r = U_r^T X, \quad (10)$$

where $U_r = [\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_r]$ and $1 \leq r \leq \min\{m, n\}$. In general, r would be much smaller than m or n , where m is the number of words in the dictionary and n is the number of documents. In fact, the transformation preserves the maximum variance of the data points for the case of projecting the data points in a r dimensional space and the reconstruction error, i.e., $\|UT - U_r T_r\|^2$, is minimized.

Actually, the new space basis can be obtained from the singular value decomposition (SVD) of X . Suppose that the SVD of X is

$$X = U\Sigma W^T, \quad (11)$$

where U is a $m \times m$ matrix in which the column vectors are orthonormal, Σ is a diagonal matrix and W is a $n \times n$ matrix, in which the column vectors are also orthonormal. From Equation (7), the eigenvalue decomposition of XX^T can be derived as

$$XX^T = U\Lambda U^T = U\Sigma W^T W \Sigma^T U^T, \quad (12)$$

where Λ is a diagonal matrix with the eigenvalues on the diagonal and $\Lambda = \Sigma\Sigma^T$. Note that Λ and Σ are diagonal matrices. Denote λ_i and σ_i as the i th term on the diagonal of matrix Λ and Σ , respectively. Thus, $\lambda_i = \sigma_i^2$ and the eigenvectors of XX^T are the left singular vectors of X . Rearrange the SVD of X as

$$U^T X = \Sigma W^T. \quad (13)$$

Obviously, the coordinates of the data points after projecting to the new space can be obtained by ΣW^T from the SVD of X . Analogously to reducing the dimensions of the index, the coordinates in the low dimensional space are given by

$$U_r^T X = \Sigma_r W_r^T. \quad (14)$$

Note that Σ_r is a $r \times r$ diagonal matrix with the first r singular values in Σ and W_r is a $n \times r$ matrix, which is composed of the first r columns of W .

4.2. Document Index and Query Vector Condensing

To do keyword search in the original space, one can construct a query vector and do the inner product of the query vector and the document indexes, as in Equation (2). The inner product can be an indicator for the similarity of the query associated to a particular document. In fact, the result of the inner product of the projected query vector and the document indexes in the new space is equal to that in the original space, as shown in the following equation:

$$(U^T \mathbf{q}) \cdot (U^T \mathbf{a}_j) = \mathbf{q}^T U U^T \mathbf{a}_j = \mathbf{q} \cdot \mathbf{a}_j, \quad (15)$$

where U is the basis of the new space, $\mathbf{q}$ is the query vector, and $\mathbf{a}_j$ is the index of the j th document. The benefit of the query in the new space is that the dimensions of the indexes can be significantly reduced with relatively small information loss. Therefore, the search efficiency can be increased, and the space can be saved. It is noted that by Equation (13), the projection of $\mathbf{a}_j$ can be derived as follows:

$$U^T \mathbf{a}_j = U^T (\mathbf{x}_j + \bar{\mathbf{a}}) = (\Sigma W^T)_j + U^T \bar{\mathbf{a}}. \quad (16)$$

Thus, the projection of $\mathbf{a}_j$ can be obtained by shifting $(\Sigma W^T)_j$ with $U^T \bar{\mathbf{a}}$, as in Equation (16). Without directly doing the projection $U^T A$, the computation time can be saved, since ΣW^T can be obtained by the SVD of X without further computation.

To achieve the search by condensed indexes, the data owner will generate the condensed document index for each document by projecting $\mathbf{a}_j$ to the low dimensional space spanned by the first r principal components. Let $\mathbf{h}_j$ be the condensed index of document f_j . The condensed index is given by

$$\mathbf{h}_j = U_r^T \mathbf{a}_j = (\Sigma_r W_r^T)_j + U_r^T \bar{\mathbf{a}}. \quad (17)$$

Analogously, the query vector has to be mapped to the low dimensional space. Let $\mathbf{g}$ be the condensed query vector, which is given by

$$\mathbf{g} = U_r^T \mathbf{q}. \quad (18)$$

4.3. Encrypted Search on the Cloud

To do encrypted search on the cloud, the condensed index is incorporated to the encryption scheme proposed in [10]. For completeness, the scheme is briefly described as follows. The data owner would generate the secret key set $SK = \{\mathbf{s}, M_1, M_2\}$, where $\mathbf{s}$ is a r -bit binary vector with 0 and 1 generated randomly, and M_1 and M_2 are $r \times r$ invertible matrices. Let $\mathbf{h}$ be a condensed document index generated as in Equation (17). To generate the encrypted document indexes, the condensed index $\mathbf{h}$ is first split by $\mathbf{s}$ to $\mathbf{h}'$ and $\mathbf{h}''$ as follows:

$$\begin{cases} \text{if } s_k = 1, & \mathbf{h}'_k = \mathbf{h}''_k = \mathbf{h}_k \\ \text{Otherwise,} & \mathbf{h}'_k + \mathbf{h}''_k = \mathbf{h}_k \end{cases}, \forall 1 \leq k \leq r, \quad (19)$$

where s_k , $\mathbf{h}_k$, $\mathbf{h}'_k$, and $\mathbf{h}''_k$ are the k th element of $\mathbf{s}$, $\mathbf{h}$, $\mathbf{h}'$, and $\mathbf{h}''$, respectively. Essentially, if s_k is 1, $\mathbf{h}'_k$ and $\mathbf{h}''_k$ are equal to $\mathbf{h}_k$; otherwise, $\mathbf{h}'_k$ and $\mathbf{h}''_k$ are randomly set such that $\mathbf{h}'_k + \mathbf{h}''_k = \mathbf{h}_k$. The encrypted document index is then given by

$$\hat{\mathbf{h}} = \{M_1^T \mathbf{h}', M_2^T \mathbf{h}''\}. \quad (20)$$

The documents are also encrypted by a certain symmetric encryption scheme. The data owner stores the encrypted documents and the associated encrypted document indexes on the cloud server.

When a data user proposes a search, the user sends the keywords to the data owner. The data owner will generate the condensed query vector and encrypt it to build the trapdoor. Suppose that the condensed query vector $\mathbf{g}$ is generated as in Equation (18). Similarly, $\mathbf{g}$ is split into two vectors $\mathbf{g}'$ and $\mathbf{g}''$ as follows.

$$\begin{cases} \text{if } s_k = 1, & \mathbf{g}'_k + \mathbf{g}''_k = \mathbf{g}_k \\ \text{Otherwise,} & \mathbf{g}'_k = \mathbf{g}''_k = \mathbf{g}_k \end{cases}, \forall 1 \leq k \leq r. \quad (21)$$

where s_k , $\mathbf{g}_k$, $\mathbf{g}'_k$, and $\mathbf{g}''_k$ are the k th element of $\mathbf{s}$, $\mathbf{g}$, $\mathbf{g}'$, and $\mathbf{g}''$, respectively. The trapdoor is generated by

$$\hat{\mathbf{g}} = \{M_1^{-1} \mathbf{g}', M_2^{-1} \mathbf{g}''\}. \quad (22)$$

The trapdoor is then given to the data user and transmitted to the cloud server. Note that, to further obfuscate the query, a vector with small random numbers can be added to the trapdoor for every query, so the cloud server will receive different trapdoor, even for the same query keywords as in [10].

After the data user sends the trapdoor to the cloud server to request a search, the cloud server performs the following computation to calculate the relevance score of the query and the documents.

$$RScore = \hat{\mathbf{h}}_j \cdot \hat{\mathbf{g}} \quad (23)$$

$$\begin{aligned} &= \{M_1^T \mathbf{h}'_j, M_2^T \mathbf{h}''_j\} \cdot \{M_1^{-1} \mathbf{g}', M_2^{-1} \mathbf{g}''\} \\ &= \mathbf{h}'_j \cdot \mathbf{g}' + \mathbf{h}''_j \cdot \mathbf{g}'' \\ &= \mathbf{h}_j \cdot \mathbf{g} \end{aligned}, \forall 1 \leq j \leq n \quad (24)$$

The *RScore* is an approximation of the relevance computed by Equation (2). However, in general, the higher the *RScore*, the better the corresponding document fits the user's request. Finally, the cloud server returns the k documents with the highest *RScore*.

Based on the encryption scheme, although the cloud server only knows the ciphertext of the query vector and the document indexes, the results are equivalent to directly calculating the inner product in plain text. However, the size of the document indexes and the trapdoor would be much smaller than those used by conventional schemes.

5. Simulations

A real dataset, which is the BBC Dataset [20], is adopted in the simulations to evaluate the performance of the proposed CIMS scheme. The dataset consists of documents in typical news areas that were collected from 2004 to 2005. There are 2250 documents in the dataset. The proposed scheme is implemented on Windows 10 with an Intel Core(TM) i7 CPU running at 3.40 GHz and 16 GB RAM. The search quality and efficiency of the proposed scheme are evaluated.

5.1. Search Quality

The documents in the dataset are first manipulated to remove the stop words and punctuation characters. A total of 26,685 words are collected from the documents to build

the dictionary. In other words, each document index contains 26,685 terms. The TF-IDF of each term in each document is first calculated. Then, the document indexes are collected to form the keyword-document matrix A . PCA is then applied on X , which is the mean-centered matrix shifted from A by the column mean. According to the results of PCA, the proposed CIMS scheme omits the dimensions which contain less information about the original document indexes and generates the condensed index for each document. Since some dimensions of the index are eliminated, the precision of the search results is first investigated. The precision is defined as follows:

$$precision = \frac{|F_A \cap F_R|}{|F_A|}, \quad (25)$$

where F_A is the set of documents returned from the cloud server when using the original long document indexes to perform the search, and F_R is the set of returned documents when using the condensed document indexes to conduct the search. Essentially, the precision characterizes the difference of the results before and after using the proposed CIMS scheme.

Figure 2 shows the simulation results for precision evaluation. For each run of the search, two words are randomly picked from the dictionary to form the query vector. The simulation uses the original indexes and the condensed indexes to perform the search and evaluates the precision. The number of returned documents from the server is set to be ten and twenty, which are denoted as $k = 10$ and $k = 20$ in the figure. Each point in the figure is an average of 100 queries. From the figure, the precision increases as the length of the condensed document index increases. The precision can reach near 100% when the length of the condensed document index is 1500, which indicates that the returned documents are almost the same as that using the original indexes. Note that the original length of the document indexes is 26,685. Obviously, the length of the index can be reduced dramatically.

Figure 2. The effects on different value of r .

5.2. Efficiency

The storage consumption of CIMS is evaluated and compared to the recent work of MRSE scheme in [10]. The MRSE scheme uses the original document index for the search. Figure 3 shows the comparison for the storage required for the document index. Each element in the document index is assumed to be four bytes. In the simulations, documents are randomly chosen from the dataset to form the simulated corpus. The number of words in the dictionary would increase as the number of documents in the simulated corpus increases. The number of words in the dictionary collected from the simulated corpus is listed in Table 1. Note that the number of words in the dictionary is the length of the original document index. In CIMS, the largest condensed index size r is $\min\{m, n\}$, where m is the number of words in the dictionary and n is the number of documents in the corpus. Therefore, there is no data for the cases ($r = 2000, n = 1500$) and ($r = 2000, n = 1750$).

From the figure, CIMS uses less than 1/10 of the storage size for the document indexes compared to MRSE. Obviously, CIMS requires less storage size than MRSE, and is expected to have less communication overhead. It is noted that, from Figure 2, the precision is almost 100% when the condensed index is greater than 1500. The search results are almost not affected by the condensed document index, but the space and communication overhead can be dramatically saved.

Figure 3. Storage efficiency.

Table 1. The number of words in dictionary.

Corpus size	500	1000	1500	1750	2000	2225
Words in dictionary	10,081	17,153	21,077	22,896	25,185	26,685

Figure 4 shows the comparison of the search time of CIMS and MRSE. Since the length of document indexes of CIMS is much smaller than that of MRSE, the computation time for doing the search of CIMS can be reduced. In the simulations, two words are randomly selected to generate the query vector and the search operations are performed in the associated corpus to find the documents. Each data point in the figure is an average of 100 runs of the search. From the figure, the searching time increases with the number of documents in the corpus. However, the search time of CIMS is less than 1/10 of that required by MRSE when the corpus size is greater than 2000. CIMS shows a very significant improvement on the search time.

Figure 4. Comparison of search time.

5.3. Preprocessing Evaluations

Compared to MRSE, one extra operation of CIMS is to do the PCA. Figure 5 shows the time of doing the PCA in CIMS. Frankly speaking, PCA requires quite a lot of time for the computation, and the time increased as the size of the corpus increased. However, PCA is only performed in the initial phase for document index generation. No further computations are required for the online search. It would not affect the user experience for the system.

Figure 5. Time of doing PCA.

Figure 6 shows the time for doing the encryption of the document indexes. Recall that the encryption operations include splitting the index to two vectors and multiplying the encryption matrices to blur the data. From the results, the required time increases with the length of the index, i.e., the r values, and the number of documents. Again, the encryption computation only occurs in the initial phase before outsourcing the documents to the cloud. Although the time increases with the size of the corpus, it would not affect the search time that users will experience online.

Figure 6. Time for doing the encryption.

The time for trapdoor generation is shown in Figure 7. In the simulations, two words are randomly selected from the dictionary to generate the query vector, which is then projected to the new space to generate the condensed query vector. The condensed query vector is then processed to generate the trapdoor. Note that the length of the condensed query vector is the same as the condensed document index, which is equal to r . From the figure, the trapdoor generation time increases with the length of the condensed query

vector. The number of documents in the corpus would not affect the query vector length, but the search time.

Figure 7. Time for generating the trapdoor.

Figure 8 illustrates the effects on different numbers of keywords in a query. The number of documents in the corpus is 2225. Since the length of the query vector is controlled by the value of r , when r increases, the searching time also increases. The CIMS scheme allows the data user to propose multiple keywords in a query. However, the search time is not affected by the number of keywords in the query, since all the keywords are included in one query vector.

Figure 8. Search time while changing the number of keywords in a query.

6. Conclusions

Cloud computing is widely used in recent years. The convenience of cloud storages has caused more and more data moved from local computers to the cloud. An efficient data search method while protecting the security and privacy of users is required. In this paper, a multi-keyword ranked search with condensed index for encrypted cloud data is proposed. From the simulation results, the document indexes can be dramatically reduced without affecting the search quality. Moreover, with the condensed document indexes, the search efficiency can also be improved in terms of time and space. There are many challenges for searching over encrypted data in the cloud. Further studies on the searchable encryption and the flexibility of keyword search are worth being explored.

Author Contributions: Conceptualization, T.-L.C. and W.-N.S.; methodology, T.-L.C. and W.-N.S.; software, W.-N.S.; validation, W.-N.S.; formal analysis, W.-N.S.; investigation, T.-L.C.; resources, T.-L.C.; data curation, W.-N.S.; writing—original draft preparation, W.-N.S.; writing—review and editing, T.-L.C.; visualization, W.-N.S.; supervision, T.-L.C.; project administration, T.-L.C.; funding acquisition, T.-L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Ministry of Science and Technology of Taiwan under grant MOST 109-2221-E-011-106.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 14–17 May 2000; pp. 44–55.
2. Goh, E.J. Secure Indexes. In Cryptology ePrint Archive. 2003. pp. 1–18. Available online: <http://eprint.iacr.org/2003/216> (accessed on 27 November 2021).
3. Golle, P.; Staddon, J.; Waters, B. Secure conjunctive keyword search over encrypted data. *Appl. Cryptogr. Netw. Secur.* **2004**, *3089*, 31–45.
4. Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G. Public key encryption with keyword search. In *Eurocrypt*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3027, pp. 506–522.
5. Ballard, L.; Kamara, S.; Monrose, F. Achieving efficient conjunctive keyword searches over encrypted data. In Proceedings of the International Conference on Information and Communications Security, Beijing, China, 10–13 December 2005; Volume 5, pp. 414–426.
6. Wang, C.; Cao, N.; Li, J.; Ren, K.; Lou, W. Secure ranked keyword search over encrypted cloud data. In Proceedings of the International Conference on Distributed Computing Systems, Genoa, Italy, 21–25 June 2010; pp. 253–262.
7. Curtmola, R.; Garay, J.; Kamara, S.; Ostrovsky, R. Searchable symmetric encryption: Improved definitions and efficient constructions. *J. Comput. Secur.* **2011**, *19*, 895–934. [\[CrossRef\]](#)
8. Chang, Y.C.; Mitzenmacher, M. Privacy preserving keyword searches on remote encrypted data. In Proceedings of the ACNS: International Conference on Applied Cryptography and Network Security, New York, NY, USA, 7–10 June 2005; Volume 5, pp. 442–455.
9. Hwang, Y.H.; Lee, P.J. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In Proceedings of the International Conference on Pairing-Based Cryptography, Tokyo, Japan, 2–4 July 2007; pp. 2–22.
10. 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.* **2014**, *25*, 222–233. [\[CrossRef\]](#)
11. 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.* **2016**, *27*, 340–352. [\[CrossRef\]](#)
12. Boneh, D.; Waters, B. Conjunctive, Subset, and Range Queries on Encrypted Data. *Theory Cryptogr.* **2007**, *4392*, 535–554.
13. Zhang, B.; Zhang, F. An efficient public key encryption with conjunctive-subset keywords search. *J. Netw. Comput. Appl.* **2011**, *34*, 262–267. [\[CrossRef\]](#)
14. Katz, J.; Sahai, A.; Waters, B. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *J. Cryptol.* **2013**, *26*, 1–34. [\[CrossRef\]](#)
15. Shen, E.; Shi, E.; Waters, B. Predicate privacy in encryption systems. In Proceedings of the TCC: Theory of Cryptography Conference, San Francisco, CA, USA, 15–17 March 2009; Volume 5444, pp. 457–473.
16. Li, J.; Wang, Q.; Wang, C.; Cao, N.; Ren, K.; Lou, W. Fuzzy keyword search over encrypted data in cloud computing. In Proceedings of the IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010.
17. Chuah, M.; Hu, W. Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data. In Proceedings of the International Conference on Distributed Computing Systems, Minneapolis, MN, USA, 20–24 June 2011; pp. 273–281.
18. Wang, B.; Yu, S.; Lou, W.; Hou, Y.T. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In Proceedings of the IEEE INFOCOM 2014-IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014; pp. 2112–2120.
19. 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.* **2016**, *11*, 2706–2716. [\[CrossRef\]](#)
20. Greene, D.; Cunningham, P. Practical solutions to the problem of diagonal dominance in kernel document clustering. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 377–384.