World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:8, No:7, 2014

# Searchable Encryption in Cloud Storage

Ren-Junn Hwang, Chung-Chien Lu, Jain-Shing Wu

***Abstract*—**Cloud outsource storage is one of important services in cloud computing. Cloud users upload data to cloud servers to reduce the cost of managing data and maintaining hardware and software. To ensure data confidentiality, users can encrypt their files before uploading them to a cloud system. However, retrieving the target file from the encrypted files exactly is difficult for cloud server. This study proposes a protocol for performing multikeyword searches for encrypted cloud data by applying k-nearest neighbor technology. The protocol ranks the relevance scores of encrypted files and keywords, and prevents cloud servers from learning search keywords submitted by a cloud user. To reduce the costs of file transfer communication, the cloud server returns encrypted files in order of relevance. Moreover, when a cloud user inputs an incorrect keyword and the number of wrong alphabet does not exceed a given threshold; the user still can retrieve the target files from cloud server. In addition, the proposed scheme satisfies security requirements for outsourced data storage.

***Keywords*—**Fault-tolerance search, multi-keywords search, outsource storage, ranked search, searchable encryption.



## I. Introduction

SONG et al. [1] proposed a searchable encryption scheme based on a symmetric key. The scheme involved applying a two-layer encryption construction to encrypt each word of the files, and the computational overhead for performing a search was relative to the file size. Goh et al. [2] proposed an index searching algorithm based on a bloom filter. Their scheme reduced the computational overhead of loading an index and searching for files. To improve that scheme, Chang et al. [3] proposed an enhanced scheme. Curtmola et al. [4] constructed an encrypted index searching algorithm based on a hash table, in which each entry comprised trapdoor and encrypted file identifiers. Bonch et al. [5] proposed a searchable encryption algorithm based on asymmetric cryptography. The scheme involved employing a public key for encrypting and uploading data to a cloud server, but only an authorized user could use a private key to search the encrypted files. Moreover, the keyword search function of these schemes [1]-[5] supported only one keyword per search.

Golle et al. [6] proposed a conjunctive keyword search scheme for encrypted data. Conjunctive keyword searches are conducted to retrieve files that contain all keywords. Compared with the schemes [1]-[5], Golle et al.'s conjunctive keyword search scheme is more accurate. Katz et al. [7] proposed a predicate encryption scheme for encrypted data that supported conjunctive and disjunctive keyword searches. A disjunctive keyword search can be conducted to retrieve files containing a subset of keywords. The scheme proposed by Katz et al. offers more flexibility in performing keyword searches. However, because the volume of data in cloud servers is increasing, even when a cloud user performs a multikeyword search, the query results typically involve a substantial number of files. The following challenges exist [8]: (1) to identify the most relevant file(s), a cloud user must decrypt the returned files; thus, the computational burden is on the user; and (2) because cloud servers return many files following a keyword search, the network communication cost is considerably high. Moreover, some of the files are unrelated to the cloud user's requirements. However, the user must pay all fees and costs associated with the returned files. Consequently, the pay-per-use principle of cloud computing is violated.

Wang et al. [9] proposed a secure ranked search for encrypted data in a cloud system. A ranked search is a keyword search that returns files based on a ranked relevance value. A ranked search can reduce the time cost of decrypting the files and communication cost of returning them. Wang et al. used keyword frequency and file size to calculate the relevance value between keywords and files. However, this scheme supported only one keyword per query. Cao et al. [9] proposed a ranked multikeyword search scheme for encrypted cloud data. Their scheme further enhanced the accuracy of the keyword search results by requiring cloud users to input exact keywords to retrieve a specific file. However, users may input an incorrect word when performing a keyword search.

Wang et al. [11] proposed another scheme to support secure similarity search in searchable encryption. When a cloud user inputs an incorrect keyword, their scheme can differentiate between the wrong and right keyword. Within the scope of that difference, it can perform a keyword search to identify matching files and return them to the user. However, their scheme supported only one keyword per search, and it did not support ranked keyword searches.

These studies provide limited keyword search functionality for cloud storage services. Thus, service providers must implement a complete secure search scheme to promote their services. This study proposes a scheme for performing ranked multikeyword searches with fault tolerance in cloud storage systems. The proposed scheme uses similar keyword sets to perform a similarity search, and a secure k-nearest neighbor (kNN) scheme to perform a ranked multikeyword search. Moreover, the proposed scheme is fault tolerant to account for cloud users inputting an incorrect keyword, and still involves performing a file search. When the files are located, they are assigned an associated correlation value. It transfers the encrypted file one by one based on the order of the relevance

Ren-Junn Hwang and Chung-Chien Lu are with the Department of Computer Science and Information Engineering, Tamkang University, New Taipei City, Taiwan (e-mail: junhwang@ms35.hinet.net, 600420078@s00.tku.edu.tw, respectively).

Jain-Shing Wuis with the Institute for Information Industry,Taipei, Taiwan(e-mail:jsw@iii.org.tw).

This work was partially supported by the Ministry of Science and Technology, Taiwan, under grants no. NSC102-2221-E-032-022.

score and stops when the cloud user already get his target files. Thus, the proposed scheme reduces the communication cost of transferring files, and the user's computational cost of decrypting the files is low. More importantly, after integrating these functions, the security is robust. Table I defines the notations of this paper.

TABLE I
NOTATION

| Notations | Meaning |
|---|---|
| $F$ | The set of $m$plaintext files, denoted as $F=(f_1, f_2,\ldots, f_m)$ |
| $C$ | The set of $m$ encryption files, denoted as $C=(c_1, c_2,\ldots, c_m)$ |
| $W$ | The set of $n$ distinct keywords, denoted as $W=(w_1, w_2,\ldots, w_n)$ |
| $I$ | Index |
| $A_j^d$ | Similar keyword set of $w_j$ and $d$ is threshold. Each key word $w'$ in $A_j^d$ satisfies $(w_j, w')\leq d$. |
| $A^d$ | All similar keyword set, denoted as $A^d=(A_1^d A_2^d,\ldots,A_n^d)$ |
| $U$ | Dummy keyword set of $m$ files, denoted as $U=(U_1, U_2,\ldots,U_m)$ |
| $\phi(\cdot)$、$\delta(\cdot)$ | pseudo-random functions defined to be $\{0,1\}^s\times\{0,1\}^*\rightarrow\{0,1\}^l$, where $s$ is the bit length of key. |
| $\pi(\cdot)$ | collision resistance hash function defined to be $\{0,1\}^s\times\{0,1\}^*\rightarrow\{0,1\}^g$, where $\{0,1\}^s$ is key. |

## II. The Proposed Scheme

This paper presents a system model comprising a cloud user and a cloud server. The user encrypts the data before uploading them to the server. To search the encrypted data, the user constructs and uploads an index of his or her encrypted data to the server. To perform a search, the user must be authorized by the server. The user enters keywords to generate a trapdoor when the search request is sent to the cloud server. The server computes the ranked score based on the index and trapdoor, and then returns highly correlated data to the user.

The threat model was developed based on the assumption that a server is honest but curious. In other words, the cloud server follows the proposed protocol, although it may analyze some queries and search results to acquire information from the outsourced data.

The proposed scheme supports ranked multikeyword searches, which tolerate queries involving incorrect keywords. Unlike many previous search schemes that support only some functions, the proposed scheme contains upload, trapdoor, and search phases. During the upload phase (Section A), a cloud user constructs an index and uploads an encrypted file and the index to a cloud server. The server stores the encrypted file and index only. No other user can infer the correlation between each keyword and file based on the index. In the trapdoor phase (Section B), the user generates a specific trapdoor based on his or her search keywords when querying the server. No other user can learn any information on the search keywords from the trapdoor. During the search phase (Section C), a server computes the margin between the trapdoor and index of each encrypted file. The server does not learn any information from the search keywords in the index or query, although the server can determine the correlation (i.e., relevance) between a user query and each file. Subsequently, the server incrementally returns the encrypted files in order of relevance to the user. When the user receives the required file, the server stops returning files to the user. Consequently, the communication overhead from transferring files is reduced, and the user conserves computational resources by not decrypting unnecessary files.

### *A. Upload Phase*

A cloud user constructs the distinct keyword set $W$ from the set of files $F$. The user also constructs a similarity keyword for each keyword, and generates an index to store the relevance score for a given keyword and file. The ranking function defines the correlation as follows.

**Ranking function**: To evaluate the correlation between a keyword and a specific file, the ranking function in (1) is used to compute a relevance score for keyword $w_j$ and file $f_i$ which inspires from [12].

$$\text{Score}(w_j, f_i) = \frac{1}{|f_i|}(1 + ln\eta_j^i)(\ln(1+\frac{m}{\eta_j}), \quad (1)$$

where $|f_i|$ denotes the number of words in $f_i$, $\eta_j^i$ is the sum of instances in which $w_j$ appears in $f_i$, $\eta_j$ represents the number of files containing $w_j$, and $m$ is the number of files. No other user can learn the correlation between each keyword and file. Subsequently, the cloud user uploads the encrypted files and index to the cloud server.

**Step U1**Determine $n$ distinct keywords from $F$ and encrypt each $f_i$ to the encrypted file $c_i$.

**Step U2**Apply Algorithm 1[11] to construct the similar keyword set $A_j^d$ for each $w_j$, where the number of difference alphabet between $w_j$ and a given similar keyword is less than $d$, $a_{j,k}^d$ denotes the $k$-th similar keyword corresponding to $w_j$, and $|a_{j,k}^d|$ is the length of the given similar keyword.

**Algorithm 1**: CreatSimilaritySet($w_j$, $d$)
**Input**: Keyword $w_j$ and threshold distance $d$
**Output**: similar keyword set $A_j^d$
**Begin**
1:set $A_j^d=\phi$
2: **if** $d\geq 1$ **then**
3: CreateSimilaritySet($w_j$, $d$-1)
4: **if** $d=0$ **then**
5: set $A_j^d=\{w_j\}$;
6: **else**
7: **for** $k\leftarrow 1$ to $|A_j^{d-1}|$ do
8:**for** $i\leftarrow 1$ to $2\times|a_{j,k}^{d-1}|+1$ do
9: set *Variant* as $a_{j,k}^{d-1}$ ;
10: **if** $i$ is odd **then**
11: Insert * at position $\lfloor(i+1)/2\rfloor$ of *Variant*;
12: **else**
13: Replace the $\lfloor i/2\rfloor$-th character of *Variant* with *;
14: **if** *Variant* is not in $A_j^{d-1}$ **then**
15: set $A_j^d= A_j^d\cup\{Variant\}$;
**End**

**Step U3**Choose $u$ dummy keywords for each file and generate a random number for each dummy keyword, where $u = 160\varepsilon$, and $\varepsilon\in Z^+$. Each dummy keyword must not exist in

*W*. The random number is generated from a uniform distribution U(-*b*, *b*), where the mean is0 and the variance is $\sigma = b^2/3$(*b* is a random value).

**Step U4**Construct the $(n+1+u)\times h$ matrix $P_i$ for each $f_i$ to store the relevance score for given values of$w_j$ and $c_i$, where $h = \max\{|A_j^d|| \ 1 < j < n\}$. The initial value of each element in $P_i$ is 0.

**Step U4.1**Calculate the relevance score for keyword $w_j$and file $f_i$ by applying(1),and store the value in $P_i[j, 1]$

**Step U4.2** Calculate the relevance score for $a_{j,k}^d$and $f_i$ by applying (1), and store the value in $P_i[j, k+1]$.

**Step U4.3**Assigna value of 1toeach field in the $(n + 1)$th row.

**Step U4.4**Assignthe first random number from Step U3 at each field of the $(n + 2)$th row, the second random number at each field of the $(n + 3)$th row, and so on.

**Step U5**Generate the index $I_i$ of $f_i$ based on $P_i$ as follows:

**Step U5.1**Generate two $(n+1+u)\times(n+1+u)$ invertible matrices $\{M_1, M_2\}$ and the secret key $S$ as an $(n+1+u)$-bit string.

**Step U5.2**Construct two $(n+1+u)\times h$ matrices $\{P_i', P_i''\}$ based on $P_i$ and $S$. If the $j$th bit of $S$ equals 0, then both $P_i'[j, k]$ and $P_i''[j, k]$ are identical to $P_i[j, k]$. If the $j$th bit of $S$ equals 1, then $P_i'[j, k]$ and $P_i''[j, k]$ are assigned to be random numbers, and the sum of $P_i'[j, k]$ and $P_i''[j, k]$ is equal to $P_i[j, k]$.

**Step U5.3**Compute $I_i = \{M_1^T P_i', M_2^T P_i''\}$, where $M_1^T$ and $M_2^T$are transposed matrices of $M_1$ and $M_2$, respectively.

**Step U6**Upload the encrypted files $C$ and index $I = \{I_i | \ i = 1,2, \ldots, m\}$ to the cloud server and then delete both $C$ and $I$. Moreover, the cloud user stores $W$, $U$, $M_1$, $M_2$, $S$, and $A^d$.

Step U3 involves constructing random numbers to ensure the confidentiality of the keywords. Step U5 involves generating $I_i$ for each instance of $f_i$ based of the keyword matrix $P_i$; this step was inspired by the kNN technique [10]. Recovering $P_i$ from $I_i$ without knowing$M_1$, $M_2$, and $S$ is difficult for any user. In addition, only the cloud user can learn the relevance score of a given $w_j$ and $c_i$ based on $I_i$.

### *B. Trapdoor Phase*

The cloud user first constructs the $(n+1+u)\times h$ matrix $Q$ based on his or her search keywords. Element $Q[j, 1]$ of row $j$in$Q$ corresponds to the keyword $w_j$, and $Q[j, k+1]$ corresponds to the similar keyword $a_{j,k}^d$for $w_j$. The initial value of each element in $Q$ is 0. Finally, the user generates a trapdoor based on $Q$. The user queries to the cloud server only including the trapdoor exclusive the search keywords. The steps in this phase are detailed as follows.

**Step T1** Apply Algorithm 1 to construct the similar keyword set $L^d$ of the search keywords.

**Step T2** Compute $K = A^d \cap L^d$.

**Step T3** Let$Q[j, k] = 1$ if the corresponding $w_j$ is included in $K$.

**Step T4**Randomly select $u/2$ rows from the $(n+2)$th row to the $(n+1+u)$th row of $Q$, and assign a value of 1 to each element in the selected rows.

**Step T5**Multiply $Q$by$r$, and then assign the random number $t$ to each element of the $(n+1)$th row.

**Step T6**Construct two $(n+1+u)\times h$ matrices $\{Q', Q''\}$based on$Q$ and $S$. If the $j$th bit of $S$ equals 1, then both $Q'[j, k]$ and $Q''[j, k]$ are assigned to be $Q[j, k]$. If the $j$th bit of $S$ equals 0, then both $Q'[j, k]$ and $Q''[j, k]$ are assigned to be random numbers, and the sum of $Q'[j, k]$ and $Q''[j, k]$ is equal to $Q[j, k]$.

**Step T7** Compute $T = \{M_1^{-1}Q', M_2^{-1}Q''\}$,where $M_1^{-1}$ and $M_2^{-1}$are inverse matrices of $M_1$ and $M_2$ respectively.

**Step T8**Generate and submit the query with the trapdoor $T$ to the cloud server.

In Step T2, $K$ includes search keywords and their fault tolerant keywords. Each keyword of $K$ is assigned a value of 1 in its corresponding element in $Q$. However, Steps 5–7 hide the content of $Q$in the trapdoor for queries inspired by the kNN technique. It is difficult for the cloud server to learn the real search keyword based on the query. During the trapdoor phase, the cloud server cannot learn which keywords appear in the searched encrypted files, although the server receives the ranked relevance scores of the encrypted files for each query.

### *C. Search Phase*

The cloud server uses the trapdoor $T$ to determine the relevance score for a given query and encrypted file $c_i$. Subsequently, the server returns the ranked relevance score of the encrypted files to the user.

**Step S1** For $T = \{M_1^{-1}Q', M_2^{-1}Q''\}$ and $I_i = \{M_1^T P_i', M_2^T P_i''\}$ of each encrypted file $c_i$, the server retrieves $h$ vector pairs for each column of $T$ and $I_i$.

**Step S2** Compute the relevance score of $c_i$ of the query by summing the inner products of each column vector of $I_i$ and $T$ by applying (2).

$$\sum_{k=1}^{h} M_1^T \overrightarrow{P'}[k] \cdot M_1^{-1}\overrightarrow{Q'}[k] + M_2^T \overrightarrow{P''}[k] \cdot M_2^{-1}\overrightarrow{Q''}[k] \qquad (2)$$

**Step S3** Return the encrypted files with higher relevance scores to the user.

The server incrementally returns the encrypted files ranked according to relevance to the user based on Step S2. Finally, the ranked search is complete. During the search phase, the cloud server cannot learn which keyword(s) appeared in the searched encrypted files.

## III. Discussions

The proposed scheme tolerates queries in which the user inputs incorrect keywords. This study proposes a technique for constructing a similar keyword set [13]. For the keyword $w_j$ and threshold $d$, the proposed scheme generates a similar keyword set $A_j^d$, where any similar keyword $w' \in A_j^d$ satisfies $\Delta(w_j, w') \le d$. Here, $\Delta(w_j, w') \le d$ implies that the number of difference alphabet between words $w_j$ and $w'$ is lower than $d$. In addition, the proposed scheme uses a wildcard ("*") to denote any character. For example, where $d = 1$ and $w_j$ = "network", 15 similar keywords {network, *network, *etwork, n*etwork, n*twork,…,networ*, network*} exist. Conversely, 390 similar

keywords are possible when the wildcard is not used. The space requirement of the proposed scheme with the wildcard is $\binom{2\lambda+1}{d}$, where λdenotes the keyword length. Conversely, the space requirement of a similar keyword set is $\binom{2\lambda+1}{d} \times 26^d$when the wildcard scheme is not used. Thus, using the wildcard enables storage space to be saved. All tables and figures you insert in your document are only to help you gauge the size of your paper, for the convenience of the referees, and to make it easy for you to distribute preprints.

Table II presents a comparison of the functionality of the multikeyword search, fault-tolerance search, and ranked search. Cao et al.'s scheme [10] used the kNN-based scheme for performing ranked multikeyword searches, but their scheme does not support fault-tolerance searches. Consequently, when a cloud user performs a keyword search by using incorrect keywords, the user cannot retrieve the correct files. Wang et al. developed a scheme [11] that supports a similarity search that is similar to the fault-tolerance search. However, that scheme does not support a ranked search method for determining the relevance between keywords and encrypted files. Furthermore, it supports only one similar keyword per query. Table II shows that the proposed scheme is more functional than the schemes proposed by Cao et al. and Wang et al.

TABLE II
FUNCTION COMPARISON

| | Multi-keyword search | Ranked search | Fault tolerance search (similar search) |
|---|---|---|---|
| Cao et al.'s scheme [10] | O | O | X |
| Wang et al.'s scheme [11] | X | X | O |
| The proposed scheme | O | O | O |

## IV. CONCLUSION

Cloud outsourced storage service reduces the hardware and software maintenance costs of the cloud user. The outsourced storage server is responsible for data management and access control. To ensure the confidentiality of uploaded data, cloud users encrypt their data before uploading them to the cloud server. This study proposes a ranked fault-tolerant multikeyword search scheme for outsourced cloud storage services to support users accessing encrypted data on a cloud server. The multikeyword search in the proposed scheme provides fault tolerance in case users input incorrect keywords. The similarity search is achieved. In addition, to enhance system efficiency, the proposed scheme uses inner product operations to evaluate the relevance scores of the files and search keywords. The cloud server returns the files ranked from highest to lowest, and relieves the network communication costs and minimizes the number of files for the user to decrypt. Thus, the ranked search is achieved. The proposed scheme is a ranked search scheme featuring fault tolerance and satisfactory security.

## REFERENCES

[1] D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," in *Proceedings of IEEE Symposium on Security and Privacy'00*, 2000, pp. 44-55.
[2] E.-J. Goh, "Secure Indexes," Cryptology ePrint Archive, 2003, [Online]. Available: http://eprint.iacr.org/2003/216.
[3] Y.-C. Chang and M. Mitzenmacher, "Privacy Preserving Keyword Searches on Remote Encrypted Data," *Applied Cryptography and Network Security*, LNCS 3531, Springer-Verlag, 2005, pp. 442-445.
[4] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," in *Proceedings of 13 th ACM Conference on Computer and Communications Security'06*, 2006, pp.79-88.
[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," *Advances in Cryptology - EUROCRYP 2004*, LNCS 3027, Springer-Verlag, 2004, pp.506-522.
[6] P. Golle, J. Staddon, and B. Waters, "Secure Conjunctive Keyword Search over Encrypted Data," *Applied Cryptography and Network Security*, LNCS 3089, Springer-Verlag, 2004, pp. 31–45.
[7] J. Katz, A. Sahai, and B. Waters, "Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products," *Advances in Cryptology – EUROCRYPT 2008*, LNCS 4965, Springer-Verlag, 2008, pp. 146-162.
[8] C. Wang, N. Cao, K. Ren and W. Lou " Enable Secure and Efficient Ranked Keyword Search over Encrypted Cloud Data" *IEEE Transactions on Parallel and Distributed Systems*, Volume 23, Issue 8, pp.1467-1479, Aug. 2012.
[9] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," in *Proceedings of IEEE International Conference on Computer Communications'11*, 2011, pp. 829-837.
[10] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proceedings of ACM SIGMOD International Conference on Management Data'09*, 2009, pp. 139-152.
[11] C. Wang, K. Ren, S. Yu, and K. Urs " Achieving Usable and Privacy-Assured Similarity Search over Outsourced Cloud Data" in *proceedings of IEEE International Conference on Computer Communications'12*, 2012, pp.451-459.
[12] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, 1st Ed. San Francisco: Morgan Kaufmann, 1999.
[13] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy Keyword Search over Encrypted Data in Cloud Computing," in *Proceedings of IEEE International Conference on Computer Communications'10*, 2010, pp. 1–5.