pISSN 1976-913X
eISSN 2092-805X

J Inf Process Syst, Vol.9, No.2, June 2013 http://dx.doi.org/10.3745/JIPS.2013.9.2.287

# A Secure Index Management Scheme for Providing Data Sharing in Cloud Storage

Sun-Ho Lee* and Im-Yeong Lee*

**Abstract**—Cloud storage is provided as a service in order to keep pace with the increasing use of digital information. It can be used to store data via networks and various devices and is easy to access. Unlike existing removable storage, many users can use cloud storage because it has no storage capacity limit and does not require a storage medium. Cloud storage reliability has become a topic of importance, as many users employ it for saving great volumes of data. For protection against unethical administrators and attackers, a variety of cryptography systems, such as searchable encryption and proxy re-encryption, are being applied to cloud storage systems. However, the existing searchable encryption technology is inconvenient to use in a cloud storage environment where users upload their data. This is because this data is shared with others, as necessary, and the users with whom the data is shared change frequently. In this paper, we propose a searchable re-encryption scheme in which a user can safely share data with others by generating a searchable encryption index and then re-encrypt it.

**Keywords**—Searchable Encryption, Proxy Re-Encryption, Index Management, Cloud Computing, Cloud Storage

## 1. INTRODUCTION

As the volume of digital information has rapidly expanded, data storage media has also rapidly developed. For removable storage, which allows us to carry data, tape drives were first developed in 1951. Since then, the technology has evolved to develop devices such as floppy disks, optical media, flash memory cards, and now USB flash drives. Removable storage media are easy to carry, so there is a high risk of loss or theft, which may lead to the disclosure of personal information saved on the media. However, because of portability, these media are also in danger of being stolen and lost, thus increasing the risk of leakage of stored data. As network development enables the acceleration of data communication, cloud-computing services that can store data in distant storage media and that can retrieve them on a user's device have been developed. Many companies are now competitively providing free high-capacity storage services. Accord-

* This research was partially supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), which is funded by the Ministry of Education, Science and Technology (2010-0022607) and by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2012-H301-12-4008), which is supervised by the NIPA (National IT Industry Promotion Agency)

Manuscript received May 24, 2012; first revision August 2, 2012; accepted October 25, 2012.

**Corresponding Author: Im-Yeong Lee**

* Dept. of Computer Software Engineering, Soonchunhyang University, Asan, Korea ({sunho431, imylee} @sch.ac.kr)

ingly, an increasing number of people are now using cloud storage services to store their data. Storing many users' data in a system increases the possibility of the "big brother problem" and the risk of data disclosure by attackers and unethical administrators.

Data encryption may be used to tackle such problems, but it makes data access difficult. Therefore, searchable encryption systems, which can encrypt data indexes and that allow index searching without exposing data to attackers and unethical administrators, have been developed [1-11]. However, this method is difficult to apply to a cloud environment where there is frequent data sharing among users. To solve this problem, a searchable re-encryption system has been developed that re-encrypts encrypted indexes and allows users to search data for safe cloud storage data sharing without using a decryption process [12]. However, existing systems do not consider users who share data with other users and the cloud storage structure, so they handle indexes and data encryption in a single process. In fact, in cloud storage, indexes and data are stored separately. The indexes are stored on the master server, and the data is split into chunks, which are then distributed to many chunk servers. Therefore, searchable re-encryption systems are difficult to apply to a real cloud storage system. Accordingly, this study proposes a technical scheme to allow the safe and free sharing of users' cloud storage data, considering the cloud storage structure.

## 2. PRELIMINARIES

Cloud computing is used to provide IT-related functions in the form of a service. Cloud computing is largely divided into three classes: SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service). In order to ensure data safety in cloud storage, a safe storage place should be provided in the IaaS class. The service provision place in a cloud-computing environment is called the "cloud storage service." The base technology used to provide this cloud storage service is the distributed file system.

Let us explore the Google File System (GFS) and the Hadoop Distributed File System (HDFS), which are the mostly widely used distributed file systems.

### 2.1 GFS

GFS is the distributed file system that is used to provide cloud services in Google. GFS consists of a client, master server, and chunk server. The roles of each component are described below.

**Client:** this provides an interface that is similar to the file system interface and it communicates with the master and chunk servers on behalf of the application program [13].

**Master server:** this manages file system metadata such as the name space, access control information, mapping information between the file and chunk, and chunk location information. These metadata are stored in the master server memory and they quickly inform the client of data locations. Furthermore, they control the overall system operations such as creating chunk copies, adjusting the number of copies, returning unused storage space, and checking the chunk server health.

**Chunk server:** the chunk server manages chunks, which are stored data units, and supports the input and output of data requested by the client. The chunk server regularly reports heartbeat

messages to the master server. Furthermore, it detects data errors by using checksums and deletes chunks in which errors are detected. GFS operation can be fully assumed by the prior GFS component role. During file storage, the client sends the file information that is to be stored to the master server and the master server sends the chunk server location and stored file handle to the client. Then, the client divides its data into fixed size chunks and sends the divided chunks to the chunk server. When reading a file, the client searches for its data on the master server and receives the chunk server location where this data is stored. Next, it receives chunks by communicating with the chunk server and retrieves the original data by adding up these chunks (refer to Figure 1).

### 2.2 HDFS

HDFS is an open source software development project for distributed computing with reliability and extensibility and has been developed to support distributed computing frameworks. This service is used by Amazon, IBM, Yahoo, etc. and is embodied by JAVA. That is, it provides good portability between platforms because it can operate on various JAVA support servers. HDFS also largely consists of three components, which are as follows: the Client, which plays the role of reading and writing user files in the system, the same as in the GFS. The Namenode, which plays the role of managing metadata, with a concept that, is similar to that of the master server in the GFS, but it does not support access right controls, unlike the GFS. The Datanode, which plays a role that, is similar to the GFS chunk server. When deleting data in the HDFS, system loads can be reduced by deleting data at certain elapsed times after moving the data to a temporary directory (i.e., the data is not deleted immediately) [14].

### 2.3 Analysis of the Distributed File System

Various distributed file systems are used, but their structures are not very different. For fast data processing, we should focus on the structure that separates the control and data storage. The data and encrypted data search index should be stored separately in order to provide safe storage places in cloud storage using a distributed file system.

### 2.4 Bilinear Pairing

The bilinear map was originally suggested as a tool for attacking elliptical curve encryption by reducing the discrete algebra of elliptical curves to discrete algebra of finite fields, thus reducing its difficulty. However, recently, it is being used not as an attacking tool but as an encryption tool for information protection. Bilinear pairing is equivalent to a bilinear map. The terms listed below are used, as stated in this paragraph, and the theory is defined below.

**Definition 1.** The characteristics that satisfy an admissible bilinear map are as follows:

Bilinear: Define a map $e = G_1 \times G_1 \rightarrow G_2$ as bilinear if $e(aP, bP) = e(P,Q)^{ab}$ where all $P,Q \in G1$, and all $a, b \in Z$.

Non-degenerate:

The map does not relate all pairs in $G_1 \times G_1$ to the identity in $G_2$. Observe that since $G_1$ and $G_2$ are prime order groups, if P is a generator of G1, G1, then e(P,P) is a generator of $G_2$

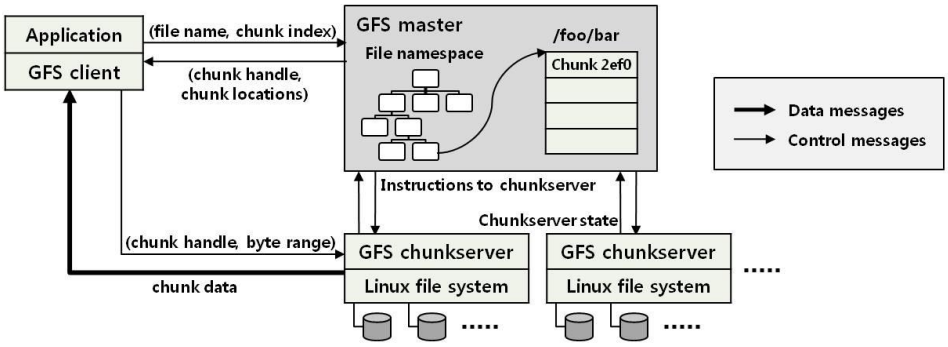


Fig. 1. Google File System Structure

Computable:

There is an efficient algorithm for computing e(P,Q) for any P, Q$\in$ $G_1$. The the definition stated below was constructed based on the bilinear map $e(aP, bQ) = e(P,bQ)^a = e(aP,Q)^b = e(P,Q)^{ab} = e(abP,Q) = e(P, abQ)$. From this map, the D-H decision problem for ellipses can be easily solved using the following equation: $e(aP, bQ) = e(cP,P) \Rightarrow ab = c$. Therefore, the the definition given below is the basis for resolving the difficulties of the bilinear map used as an encryption tool by many encryption protocols.

**Definition 2.** When elements $G_1$, P, aP, bP, cP (BDHP, Bilinear Diffie–Hellman Problem) are given, this refers to the $e(P,P)^{abc}$ calculation problem. In this research, the admissible bilinear map was used as the basis of secret number production in the key construction process between heterogeneous devices. This problem can be solved if the ellipse curve discrete mathematics problem can be solved. For example, a can be calculated from aP; then $e(P,P)^{abc}$ can be calculated through $e(bP, cP)^a$.

## 2.5 Requirements

The requirements listed below should be met to ensure safe searching and sharing under a cloud storage environment.

**Confidentiality:** data transmitted between the cloud storage server and client terminal should be identifiable only by validated users.

**Search speed:** a client who has limited system resources should be able to quickly search documents, including word files, that are stored in cloud storage systems.

**Traffic efficiency:** the communication volume should be small for network resource efficiency and energy efficiency between the client and server.

**Calculation efficiency:** a calculation efficiency should be provided for index generation, search execution, and for the safe sharing of data with other users.

**Sharing efficiency among users:** encrypted data must be retrieved from saved distant data and be protected and safely and efficiently shared with those users who use an unreliable server. Cloud service providers should only make the data that the data owner wishes to share with another user, shareable.

## 3. RELATED WORK

In this section, we present specific studies that considered sharing among users in a cloud storage environment.

### 3.1 Hwang et al.

Hwang and Lee proposed mPECK, which considers a multiple-receiver environment. Their method proceed as stated below (see Fig 2.) [6].

**KeyGen:** a pair of secret and public keys is safely sent to each user.
**mPECK:** the sender encrypts the data index with the receiver's public keys and keywords.
**Trapdoor:** the receiver generates a trapdoor with their secret key and keywords.
**Test:** using the trapdoor, the server tests whether users' keywords correspond to the data stored.

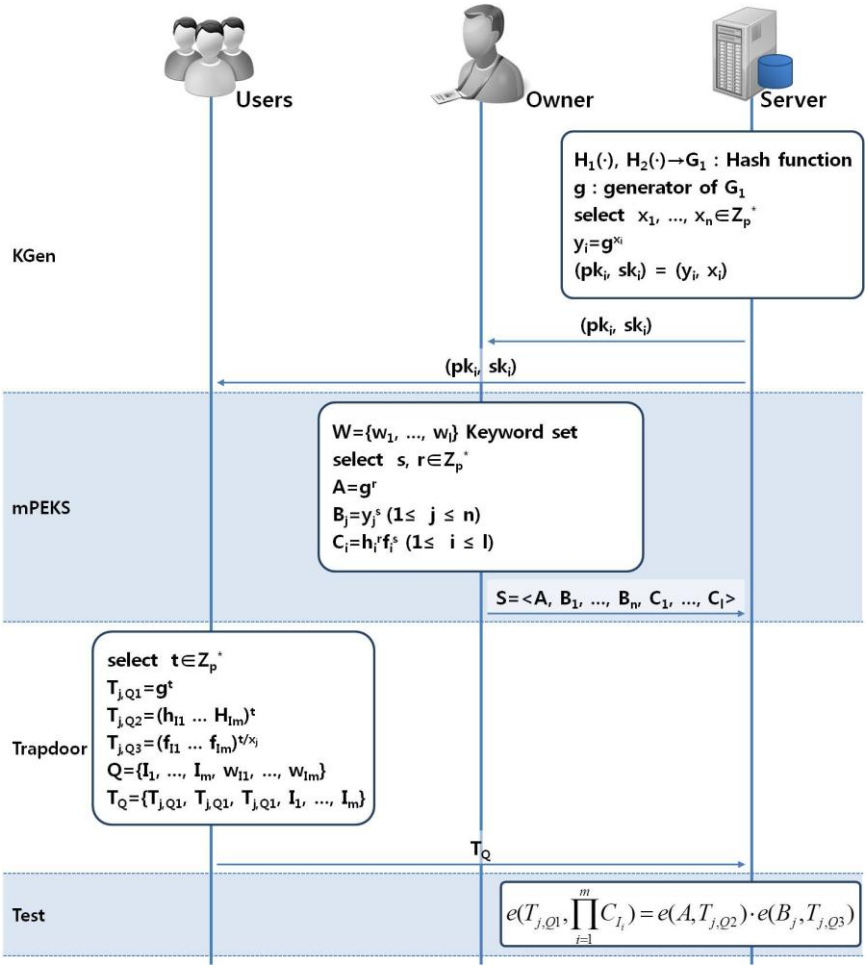


Fig. 2. Google File System Structure

In this scheme, the shared subjects are defined in advance. If the shared subject has been changed, the index needs to be re-created, which leads to operation inefficiency.

### 3.2 Bao et al.

Bao et al. adopted a UM (User Manager) to grant receiver rights to the user that generates the data. The procedure used by them is as stated below (see Fig. 3) [7].

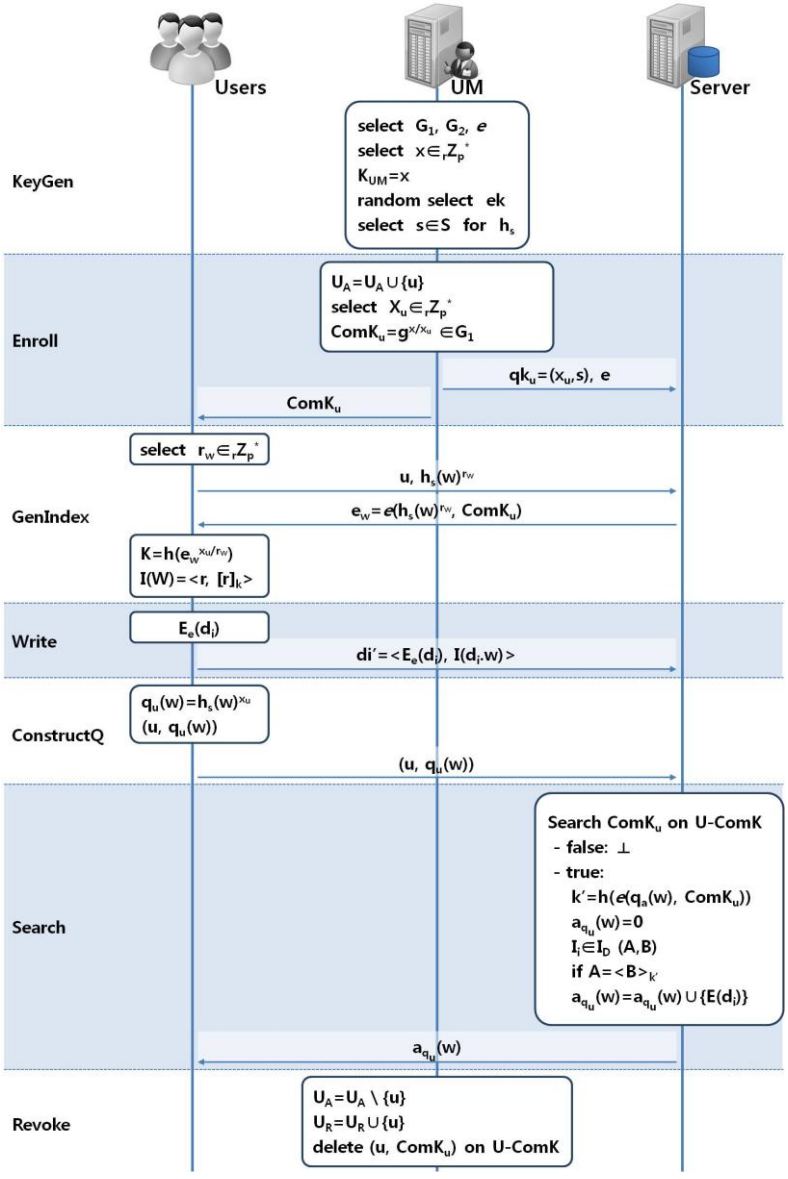


Fig. 3. Protocol flow diagram of Bao et al

**Setup:** set up the various parameters that are used in the scheme.
**Enroll:** the UM grants search rights to other users.
**GenIndex:** the user generates an index for keywords.
**Write:** the user records the encrypted data and index on the server.
**ConstructQ:** the user generates a query for a keyword search.
**Search:** the server searches data with the received query.
**Revoke: the** UM revokes the search rights of a user.

This scheme can manage the subject of shared data freely, but it incurs an additional cost for the construction of a UM server.

### 3.3 Chen et al.

Chen and Lee proposed a searchable re-encryption scheme for data sharing on unreliable storage. The procedure of this scheme is as described below (see Fig. 4)[12].

**KGen:** generate a key pair for each subject.
**Enc:** this phase is performed by the data owner to encrypt the keyword and message.
**RKGen:** a data owner takes their own private key and a user's public key as inputs, and produces the re-encryption key for sharing their own data with the user.

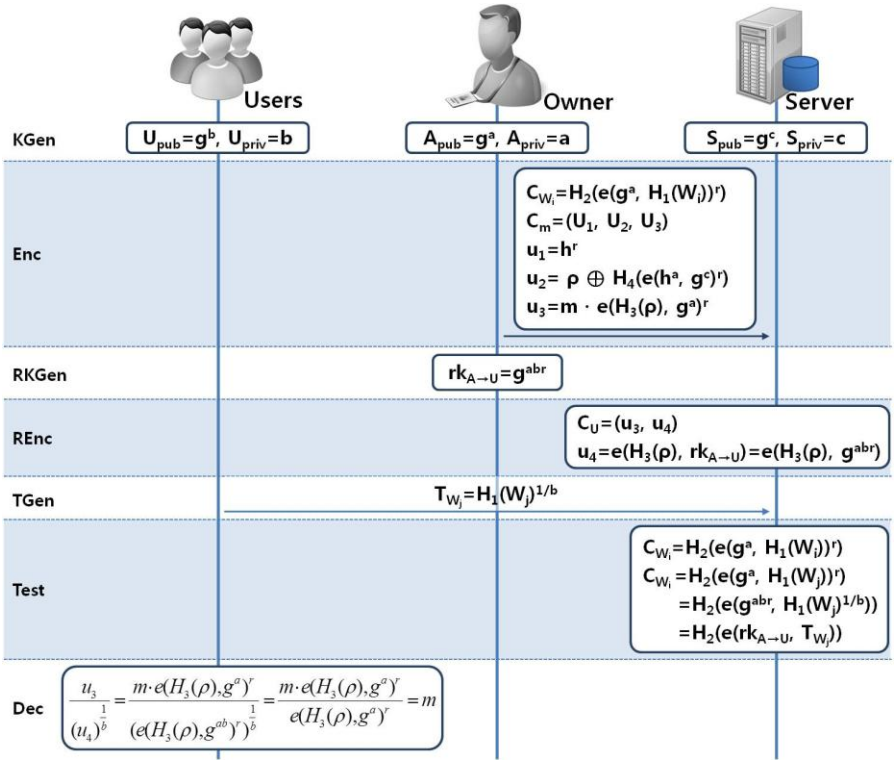


Fig. 4. Protocol flow diagram of Chen et al

**REnc:** the server takes the re-encryption key and the data owner's cipher text and produces a re-encrypted cipher text for the user.

**TGen:** the user takes their private key and a keyword as inputs, and produces a trapdoor.

**Test:** the server takes the trapdoor as an input, and finds matching data.

**Dec:** the user takes their private key and the re-encrypted cipher text as inputs, and computes the plaintext.

This scheme can add to the subject for data sharing after the data has been uploaded on cloud storage. However, it does not consider a scenario in which re-sharing the data with other users is desired. In addition, this scheme stores the encrypted data and index in one place. Therefore, it is not suitable for a cloud storage environment because the index and data are stored separately in a cloud storage structure.

## 4. PROPOSED SCHEME

In this section, the structural characteristics of a distributed file system that was analyzed beforehand are considered and the scenarios in which our schemes are operated and the roles taken in each step for satisfying requirements are defined.

### 4.1 Definitions

Detailed calculations are performed in the suggested method as described below.

**KeyGen:** the cloud storage user safely receives public key pairs created by the TA (Trust Authority) prior to using the service. The cloud storage server should not be able to know the user's private key. If the private key is leaked, the attacker can generate a trapdoor as the owner of the private key. We generated a key pair based on the Discrete Logarithm Problem (DLP).

**Enc($sk_a$, $pk_a$, w)→$E_a$:** the data owner A creates the encrypted index, $E_a$, which only A can search by inputting their own personal key $sk_a$, public key $pk_a$, and keyword w, and sends this to the master server.

**ReKeyGen($sk_a$, $pk_b$)→$rk_{a→b}$:** the data owner A creates a re-encryption key, $rk_{a→b}$, to create a data index for sharing that B can search. The re-encryption key is created with the data owner's personal key $sk_a$, and the public key $pk_b$ of the user who will be sharing the data.

**ReEnc($rk_{a→b}$, $E_a$)→$E_b$:** the master server creates a new index, $E_b$, which B can search with the re-encryption key $rk_{a→b}$ that was received from the user and the encrypted index $E_a$ of the data shared by A.

**TrapdoorGen(sk, w)→$T_w$:** to safely search data, the user creates a trapdoor, $T_w$, which does not leak the information of the keyword w being searched by them using their personal key sk. The trapdoor is safely sent to the master server. The administrator of the cloud storage should not be able to gain information from the trapdoor.

**Test(E, $T_w$)→"yes" or "no":** by using the trapdoor generated with the user's public key and search keywords, the server performs a test to confirm whether the encrypted data contains those keywords. If the cipher text has the specified keywords, the server sends a "yes" to the user, and, if not, it sends a "no." Here, the server cannot learn anything about the keywords or data.

**Dec(sk, E)→m:** the rightful owner of the encrypted data uses their private key to decrypt the

encrypted data.

### 4.2 System Parameters

**p:** prime number
**G:** cyclic additive group of order p
**$G_T$ :** cyclic multiplicative group of order p
**g:** generator of G
**e:** bilinear map, $G \times G \rightarrow G_T$
**$sk_*$:** *'s secret key
**$pk_*$:** *'s public key
**$EC_k()$:** symmetric key encryption by key k
**$DC_k()$:** symmetric key decryption by key k
$K_d$: data encryption key
**d:** data for encryption
**$w_*$:** $*^{th}$ keyword of data
**m:** plain text
**$H_1()$:** hash function, $\{0,1\}^* \rightarrow G$
**$H_2()$:** hash function, $\{0,1\}^* \rightarrow G$
**$H_3()$:** hash function, $GT \rightarrow \{0,1\}^*$
**$T_*$:** trapdoor searching keyword *
**$rk_{a \rightarrow b}$:** re-encryption key changing A's crypt to B's crypt

### 4.3 Writing Scenario

In the proposed method that considers a cloud storage structure, an encrypting index for sharing and searching is stored on the master server. We assume that each user has received a key pair before using the cloud storage services (refer to the KeyGen phase). The user encrypts the necessary keywords during the data search in order to be able to later perform their own search and sends this to the master server (refer to the Enc phase). The master server sends chunk information for data storage to the user, who then divides the data into chunks and stores them on the designated chunk server (refer to Figure 5).

**KeyGen phase**
TA generates a pair of keys and safely sends it to the cloud storage user.
$x \in Z_q$ selection
$sk = x$ setting up
$pk = g^x$ setting up

**Enc phase**
The data owner, A, generates the cipher-text that can be used for conducting a secure search.
$A = pk_a^r$ $(r \in Z_p)$
$B = e(g,g)^{ska \cdot r}$
$c_i = H_3(e(g,H_1(w_i))^r)$
$C=\{c_1, c_2, ..., c_l\}$
$D=e(g, H_2(sk))^r \cdot K_d$

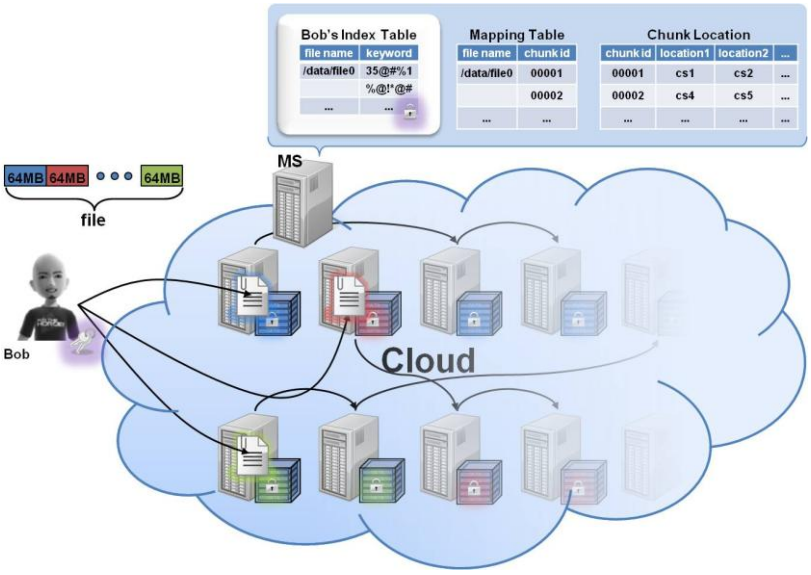


Fig. 5. Writing scenario

$E_a = (A, B, C, D)$ output as encrypted index

## 4.4 Reading Scenario

The user sends a trapdoor that can search data without exposing keyword information to the master server (refer to the TrapdoorGen phase). The master server searches for the data having the keyword using the trapdoor in the encrypted index and then sends chunk information corresponding to the data to the user (refer to the Test phase). The legitimate user decrypts the retrieved data (refer to the Dec phase). The user acquires the data by adding these up after receiving each chunk from the chunk server that stores the data (refer to Figure 6).

**TrapdoorGen phase**

The user who wishes to search the data generates a trapdoor using the keywords and their secret key.

$T_w = H_1(w)^{-sk}$

**Test phase**

To confirm whether the data contains the keywords that the user is looking for, the user performs the following tests using their public key, trapdoor, and crypt obtained from the server.

$c_i = ? H_3(e(A,T_w))$

**Dec phase**

The legitimate user acquires the data encryption key from the index, and then decrypts the encrypted data using a symmetric decryption algorithm.

$K_d = D/e(A,H_2(sk))^{-sk}$

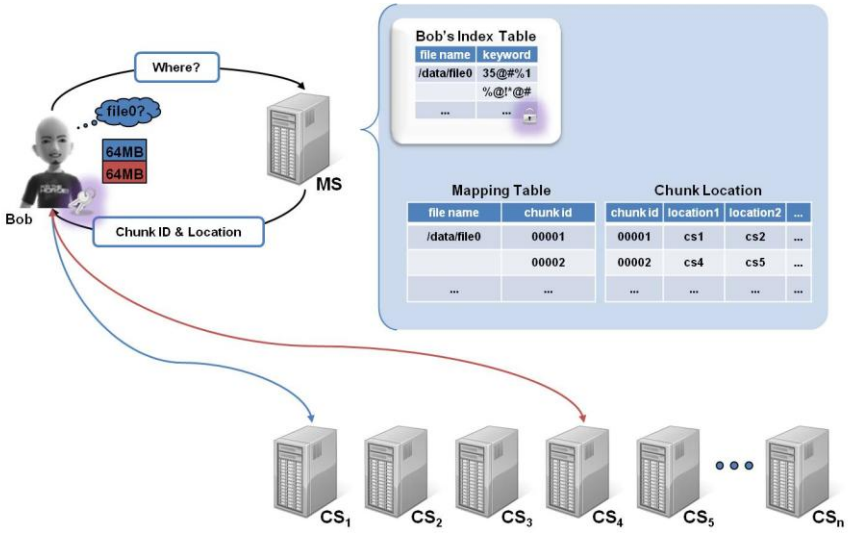


Fig. 6. Reading scenario

$c_i = D_{Kd}(e_i)$ $(i=1\sim l)$
$d=\{c_1, c_2, \ldots, c_l\}$

## 4.5 Sharing Scenario

In order to share data with the desired user and for shared users to freely share data with another user, re-encryption should be performed to allow shared users to only search the encrypted index (refer to Figure 7).

To apply proxy re-encryption and a searchable encryption scheme separately for secure data sharing in a cloud storage environment, many parameters are required, which reduces the storage volume efficiency. Therefore, we propose an algorithm that provides both functions at the same time. First, the re-encryption key for sharing the index with another user is produced and sent to the cloud storage provider by the owner of data (refer to the ReKeyGen phase). Then, the cloud storage provider re-encrypts the owner's index for the target of the data sharing (refer to the ReEnc phase). A shared (re-encrypted) data search is possible, as shown in Section 4.3. The user acquiring the data-sharing index can always search for the corresponding data using keywords and can then download it.

**ReKeyGen phase**

When the data owner wants to share their data with other users, they generate keys for re-encryption. When user A wants to share their data with user B, A generates a re-encryption key using A's secret key and B's public key, as follows:

$rk_{a\rightarrow b} = pk_b^{-ska} \bmod p$

**ReEnc phase**

The cloud storage service server performs re-encryption with the re-encryption key inputted by the user, the target crypt that is intended to be re-encrypted, and the public key, as follows:

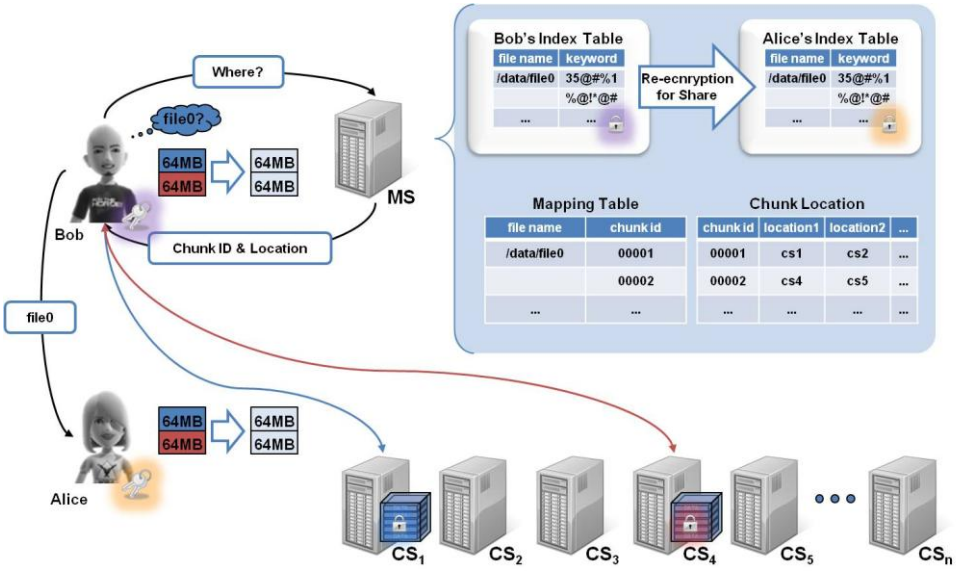


Fig. 7. Sharing scenario

$A' = A^{skb/ska}$
$B' = e(A, rk_{a \to b})$
$E_b = (A', B', C, D)$

## 5. ANALYSIS

The proposed method satisfies the requirements stated below.

**Confidentiality:** by using pairing, the proposed method makes it difficult for a malicious third party to decrypt communication contents, even if they bug the communication between the client and server.

**Search speed:** a user can check whether a document contains keywords by performing single pairing and hash calculations, which increases the searching speed.

**Traffic efficiency:** keyword search and re-encryption needs only one round of communication. Therefore, the method increases communication volume efficiency.

**Calculation efficiency:** based on the relatively simple pairing calculation, the method allows users to generate the index and search documents, and to perform re-encryption, thereby increasing calculation efficiency (see Table 1).

**Sharing efficiency among users:** our scheme allows encrypted and stored data on an unreliable distant cloud storage server to be shared safely and efficiently. In addition, the proposed method is different from existing methods, in that it does not require the shared subjects to be specified in advance, and it does not require additional devices to manage the subjects receiving shared data. Finally, if users want to re-share the data shared from the owner to other users, they only need to complete the RKGen and REnc phases in an unreliable cloud storage environment.

Table 1. Analysis of the calculation efficiency

| | Hwang | Bao | Chen | Proposed Method |
|---|---|---|---|---|
| **Calculation Volume for the Test** | 3vp | vp | vp | vp |
| **Calculation Volume for Trapdoor generation** | 3(H+e) | H+e | H+e | H+e |
| **Size of Trapdoor** | 3λ | λ | λ | λ |
| **Availability of Conjunctive keyword search** | Support | Non-support | Non-support | Non-support |
| **Notes** | - | Creation of the additional constituting cost of UM | Provision of a one-time Re-encryption | Provision of a Bidirectional Re-encryption |

v: number of document, p: pairing operation, H: Hash operation, e: exponentiation operation

## 6. CONCLUSION

The advent of cloud storage services has enabled many users to store and access data. Research on the application of searchable encryption technology to cloud storage was recently begun so as to ensure the security of data stored in cloud storage. However, most existing searchable encryption technologies are inefficient when adding data sharing objects because they are based on e-mail environments, and thus, determine the objects with which data can be shared. In a cloud storage environment, users upload data on their own and share the data in a safe manner. Therefore, indexes and data are separated. Consequently, the existing technologies are compatible with cloud storage systems. To this end, by considering such requirements in the cloud storage environment, we set up security requirements and have proposed a method for providing the following two functions simultaneously: a proxy re-encryption function and a searchable encryption function. The proposed method provides a free sharing feature with the same calculation efficiency as existing methods. It seems that search methods using multiple keywords will become important issues in ensuring the flexibility and ease of searching data in cloud storage. Therefore, in the future, it will be necessary to develop a re-encryption system where an index composed of multiple keywords of variable lengths can be encrypted and flexibly searched.

## REFERENCES

[1] D. X. Song, D. Wagner and A. Perrig, "*Practical Techniques for Searching on Encrypted Data*," *Symposium on Security and Privacy*, California, USA, May, 2000, pp.14-17.

[2] E. J. Goh, "Secure Indexes," *ePrint Crpytography Archive*, 2004.

[3] R. Curtmola, J. Garay, S. Kamara and R. Ostrovsky, "*Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions*," *Proceedings of the 13th ACM conference on Computer and communications security*, Virginia, USA, October, 2006.

[4] D. Boneh, G. Crescenzo, R. Ostrovsky and G. Persiano, "*Public Key Encryption with Keyword Search*," *Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, May, 2004.

[5] D. Boneh and B. Waters, Conjunctive, "*Subset and Range Queries on Encrypted Data*," *Proceedings of the 4th Theory of Cryptography Conference*, Amsterdam, Netherlands, Febraury, 2007.

[6] Y. H. Hwang and P. J. Lee, "*Public key encryption with conjunctive keyword search and its extension to a multi-user system*," *Proceeding of First International Conference on Pairing-Based Cryptography*, Tokyo, Japan, July, 2007.
[7] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "*Private Query on Encrypted Data in Multi-User Settings*," *Proceeding of the 4th international conference on Information security practice and experience*, Sydney, Australia, April, 2008.
[8] Kamara, S. and Lauter. K., "*Cryptographic Cloud Storage*," *Proceedings of Workshops on Financial Cryptography and Data Security*, Canary Islands, Spain, January, 2010.
[9] Ion, M., Russello, G. and Crispo, B., "*Enforcing Multi-user Access Policies to Encrypted Cloud Databases*," *International Symposium on Policies for Distributed Systems and Networks*, Trento, Italy, June, 2011.
[10] B. Zhang, and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *Journal of Network and Computer Applications*. Vol 34, No.1, 2011.
[11] Y. Yang, "*Towards Multi-user Private Keyword Search for Cloud Computing*," *Proceeding of International Conference on Cloud Computing*, Singapore, Singapore, July, 2011.
[12] Chen, X., Li, Y., "Efficient Proxy Re-encryption with Private Keyword Searching in Untrusted Storage," *I.J. Computer Network and Information Security*. Vol.3, No.2, 2011.
[13] S. Ghemawat, H. Gobioff, and S. Leung, "*The Google File System*," *Proceedings of the nineteenth ACM symposium on Operating systems principles*, Newyork, USA, December, 2003.
[14] D Borthakur, "The Hadoop Distributed File Aystem: Architecture and Design," http://svn.apache.org/repos/asf/hadoop/common/tags/release-0.16.1/docs/hdfs_design.pdf

**Sun-Ho Lee**

He received the B.S. and M.S. degrees in Depart of Computer Software Engineering from Soonchunhyang University, Korea, in 2009 and 2011, respectively. He is now a Ph.D. candidate in Department of Computer Software Engineering from Soonchunhyang University, Korea. His research interests include Searchable encryption, Secure USB flash drive, Cloud computing Security, etc.

**Im-Yeong Lee**

He received the B.S. degrees in Department of Electronic Engineering from Hongik University, Korea, in 1981 and the M.S. and Ph.D. degrees in Department of Communication Engineering from Osaka University, Japan, in 1986 and 1989, respectively. From 1989 to 1994, he had been a senior researcher at ETRI (Electronics and Telecommunications Research Institute), Korea. Now he is a professor in Department of Computer Software Engineering from Soonchunhyang University, Korea. His research interests include Cryptography, Information theory, Computer & Network security.