Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 615679, 10 pages
http://dx.doi.org/10.1155/2014/615679

Research Article

A Study of Practical Proxy Reencryption with a Keyword Search Scheme considering Cloud Storage Structure

Sun-Ho Lee and Im-Yeong Lee

Department of Computer Software Engineering, Soonchunhyang University, Asan-si, Chungcheongnam-do, Republic of Korea

Correspondence should be addressed to Im-Yeong Lee; imyle@sch.ac.kr

Received 31 August 2013; Accepted 22 October 2013; Published 12 February 2014

Academic Editors: H. Cheng and H.-E. Tseng

Data outsourcing services have emerged with the increasing use of digital information. They can be used to store data from various devices via networks that are easy to access. Unlike existing removable storage systems, storage outsourcing is available to many users because it has no storage limit and does not require a local storage medium. However, the reliability of storage outsourcing has become an important topic because many users employ it to store large volumes of data. To protect against unethical administrators and attackers, a variety of cryptography systems are used, such as searchable encryption and proxy reencryption. However, existing searchable encryption technology is inconvenient for use in storage outsourcing environments where users upload their data to be shared with others as necessary. In addition, some existing schemes are vulnerable to collusion attacks and have computing cost inefficiencies. In this paper, we analyze existing proxy re-encryption with keyword search.

1. Introduction

Network development has accelerated data communication, and data outsourcing services have been developed to store data in distant storage media, which can be retrieved by a user with various devices. Many companies are now providing competitive high-capacity storage services. Thus, an increasing number of people are using storage outsourcing services to store their data. However, the storage of sensitive data such as medical or financial information increases the development of the "Big Brother problem" and the risk of data disclosure by attackers and unethical administrators.

One scheme for protecting user data is data encryption on the data outsourcing server. However, this approach can cause difficulties during data access. Users must download all of their own data, and decryption needs to be applied to the entire dataset before the data can be searched. This can be viewed as a major disadvantage of data outsourcing. Therefore, searchable encryption systems have been developed that can encrypt data indexes to allow index searching without exposing the data to attackers and unethical administrators.

The study of searchable encryption systems began with searchable symmetric encryption (SSE) based on symmetric key cryptography as well as the development of generic cryptographic algorithms. The first construction of SSE was proposed by Song et al. [1]. Then, a new scheme using the Bloom filter schemes was proposed by Goh [2]. In order to provide faster retrieval time, an SSE scheme using an encrypted linked list scheme was announced by Curtmola et al. [3].

Next, research into searchable encryption systems based on a public-key has been actively carried out. The first public key encryption with keyword search (PEKS) using a bilinear map was proposed by Boneh et al. [4]. The PEKS scheme provides a variety of functions; for example, multiuser capability was proposed [5–11].

However, this scheme is difficult to apply in a cloud environment where there is frequent data sharing among users. To address this problem, a proxy reencryption with keyword search (PRES) system has been developed that reencrypts encrypted indexes and allows users to search during safe data storage outsourcing and sharing without the need for a decryption process [12–14].

However, some existing systems do not consider users who share data with other users or the storage outsourcing structure, which means that they handle the indexes and data encryption as a single process. In reality, the indexes and data

are stored separately during storage outsourcing. The indexes are stored on the master server, and the data are split into chunks, which are then distributed to many chunk servers. Therefore, searchable reencryption systems are difficult to apply to a real outsourced storage system. In addition, some existing schemes are vulnerable to collusion attack. Some existing schemes allow only one-hop data sharing. In reality, there is no longer any control after the data have been shared. If data need to be shared, the user has no choice other than to accept multihop reencryption. Most searchable reencryption schemes require large volumes of computing resources for data storage and sharing.

The present study examined the operation process of PRES, which operated in the same manner as the above scenario, and analyzed the consequences of a relevant scheme for collusion with an administrator of an untrusted remote storage and sharing target.

2. Preliminaries

In this section, we provide the necessary preliminary details.

2.1. Bilinear Maps. The bilinear map was proposed originally as a tool for attacking elliptical curve encryption by reducing the problem of discrete algebra on an elliptical curve to the problem of discrete algebra in a finite field, thereby reducing its complexity. However, this scheme has been used recently as an encryption tool for information protection, instead of an attacking tool. Bilinear pairing is equivalent to a bilinear map. These terms are defined and the theory is described below.

Definition 1. Characteristics that satisfy an admissible bilinear map are as follows.

(i) Bilinear: define a map $e = G \times G \rightarrow G_T$ as bilinear if $e(P^a, P^b) = e(P, Q)^{ab}$ where all $P, Q \in G$, and all a, $b \in Z$.

(ii) Nondegenerate: the map does not relate all pairs in $G \times G$ to the identity in G_T. Note that G and G_T are groups of prime order, which implies that if P is a generator of G, $e(P, P)$ is a generator of G_T.

(iii) Computable: there is an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in G$. The following definition was constructed based on the bilinear map $e(P^a, Q^b) = e(P, Q^b)^a = e(P^a, Q)^b = e(P, Q)^{ab} = e(P^{ab}, Q) = e(P, Q^{ab})$. With this map, the D-H decision problem can be solved readily for ellipses using the following equation: $e(P^a, Q^b) = e(P^c, P) \Rightarrow ab = c$. Therefore, the following is the basis for resolving the difficulties of the bilinear map, which is used as an encryption tool by many encryption protocols.

Definition 2. When the elements G, P, P^a, P^b, P^c (BDHP, Bilinear Diffie-Hellman Problem) are given, this relates to the $e(P, P)^{abc}$ calculation problem. In this study, the admissible bilinear map was used as the basis for secret number production during 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 P^a, so $e(P, P)^{abc}$ can be calculated using $e(P^b, P^c)^a$.

2.2. Existing PRES Scheme. Let us take a look at [13] proposed by Chen and Li in 2011.

2.2.1. Notation. The notation used in this scheme are as follows.

(i) q: Prime number.

(ii) G_1: Cyclic additive group of order p.

(iii) G_2: Cyclic multiplicative group of order p.

(iv) g: Generator of G.

(v) e: Bilinear map, $G_1 \times G_1 \rightarrow G_2$.

(vi) $H_1(\,)$: Hash function, $\{0,1\}^* \rightarrow G_1^*$.

(vii) $H_2(\,)$: Hash function, $G_2 \rightarrow \{0,1\}^{\log q}$.

(viii) $H_3(\,)$: Hash function, $\{0,1\}^* \rightarrow G_1^*$.

(ix) $H_4(\,)$: Hash function, $G_2 \rightarrow \{0,1\}^n$.

2.2.2. Protocol. As with most PRES schemes, the protocol of Chen et al. had a total of 7 phases: KGen, Enc, RKGen, REnc, TGen, Test, and Dec.

KGen Phase. Objects each public/private open key pairs using remote storage in the KGen stage:

$$\begin{aligned} &\text{Alice: } A_{\text{pub}} = g^a, \qquad A_{\text{priv}} = a \in Z_p \\ &\text{Bob: } B_{\text{pub}} = g^b, \qquad B_{\text{priv}} = b \in Z_p \\ &\text{Server: } S_{\text{pub}} = g^s, \qquad S_{\text{priv}} = s \in Z_p. \end{aligned} \tag{1}$$

Enc Phase. User A transmits encrypted data to remote storage S:

$$\begin{aligned} &r \in Z_q^* \\ &\rho \in \{0,1\}^n \\ &u_1 = h^r \\ &\text{Alice: } u_2 = \rho \oplus H_4\left(e\left(h^a, g^s\right)^r\right) \\ &u_3 = m \cdot e\left(H_3(\rho), g^a\right)^r \\ &C_{W_i} = H_2\left(e\left(g^a, H_1(W_i)\right)^r\right) \\ &C_m = (u_1, u_2, u_3) \\ &\text{Alice} \longrightarrow \text{Server: } C_{W_1}, C_{W_2}, \ldots, C_{W_k}, C_m. \end{aligned} \tag{2}$$

RKGen Phase. User A transmits a reencryption key to S in order to share data with B:

$$\text{Alice} \longrightarrow \text{Server: } rk_{A \rightarrow B} = g^{abr}. \tag{3}$$

REnc Phase. S reencrypts the data with the reencryption key transmitted by A.

$$\begin{aligned} \rho &= u_2 \oplus H_4\left(e(h^a, g^s)^r\right) \\ &= u_2 \oplus H_4\left(e(h^r, g^a)^s\right) \\ \text{Server:}\quad u_4 &= e\left(H_3(\rho), rk_{A\to B}\right) \\ &= e\left(H_3(\rho), g^{abr}\right) \\ C_B &= (u_3, u_4). \end{aligned} \tag{4}$$

TGen Phase. User B transmits a produced trapdoor to S in order to search the shared data from A:

$$\text{Alice} \longrightarrow \text{Server: } T_{W_j} = H_1\left(W_j\right)^{1/b}. \tag{5}$$

Test Phase. S transmits the search results after searching the data using the trapdoor sent from B.

$$\begin{aligned} \text{Server: } C_{W_i} &= ?H_2\left(e\left(rk_{A\to B}, T_{W_j}\right)\right) \\ &= H_2\left(e\left(g^{abr}, H_1\left(W_j\right)^{1/b}\right)\right) \\ &= H_2\left(e\left(g^a, H_1\left(W_j\right)^r\right)\right). \end{aligned} \tag{6}$$

Dec Phase. User B verifies the contents by decrypting the data relevant to the search results:

$$\begin{aligned} \text{Bob: } \frac{u_3}{(u_4)^{1/b}} &= \frac{m \cdot e(H_3(\rho), g^a)^r}{\left(e(H_3(\rho), g^{ab})^r\right)^{1/b}} \\ &= \frac{m \cdot e(H_3(\rho), g^a)^r}{e(H_3(\rho), g^a)^r} = m. \end{aligned} \tag{7}$$

2.2.3. An Analysis on the Protocol. We will analyze the PRES scheme proposed by Chen et al. for possible security threats.

Analysis 1: Problem of Sharing Process. In RKGen phase, A produced $rk_{A\to B} = g^{abr}$ to share his own data. This is known as producing with the similar scheme of $(g^b)^{ar}$. However, the value of r is not knowable even if the data owner is A. In the Enc phase, A produces a random r value according to different data and does not save it separately. In addition, directly deducting the r value used only in a multiplication operation from the encrypted data is not possible even if A is the data owner. In other words, A cannot produce an $rk_{A\to B} = g^{abr}$ value to reencrypt the uploaded data. In order for RKGen to be established, the r value should be opened, or A should save all r values relevant to each data set.

Analysis 2: Collusion Problem. Let us suppose that S and B are in collusion. If the r value is revealed, or all files are encrypted using the same r values, S can easily produce the reencryption key $rk'_{A\to B} = (g^a)^{br}$ by using the open key of data owner A and the personal key of colluder B. Then, a file that is not a sharing object can be reencrypted as below for B:

$$\begin{aligned} \rho &= u_2 \oplus H_4\left(e(h^a, g^s)^r\right) \\ &= u_2 \oplus H_4\left(e(h^r, g^a)^s\right) \\ \text{Server:}\quad u_4 &= e\left(H_3(\rho), rk'_{A\to B}\right) \\ &= e\left(H_3(\rho), g^{abr}\right) \\ C_B &= (u_3, u_4). \end{aligned} \tag{8}$$

According to the above scheme, unwanted sharing not just with B but with anybody that A did not want is possible.

Analysis 3: Data Encryption Problem. In Enc phase the process $u_3 = m \cdot e(H_3(\rho), g^a)^r$ is implemented to encrypt the data. In other words, a multiplicative group encrypts messages by multiplying the element and the message in an elliptic curve situation. The multiplication operation of the elliptic curve is only possible with elements of the multiplicative group. In other words, when changing a multiplicative group of plaintext to an element, obtaining a normal plaintext value is not possible during decryption in the future.

2.3. Security Requirement. The following requirements should be met to ensure safe searching and sharing in an outsourced storage environment.

(i) Confidentiality: data transmitted between the outsourced storage server and client terminal should be identifiable only by validated users.

(ii) Search speed: a client who has limited system resources should be able to search documents quickly, including word processing files, stored in outsourced storage systems. In the case where the data index structure of the existing scheme is the same as in Figure 1, and the server needs to retrieve data from all indexes to find the data containing the keyword, it is very inefficient. In addition, many previously developed search algorithms do not apply to this structure, so the storage server must perform a sequential search. In this structure, the scan speed decreases rapidly with an increasing number of documents. In order to solve this problem, the structure of the encryption index must be changed, as shown in Figure 2. If we adopt such a structure, the previously developed fastest search algorithm can be used for the data search.

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

(iv) Calculation efficiency: calculation efficiency should be provided for index generation, search execution, and safe sharing of data with other users. The previous scheme is highly inefficient for encrypting variable-length data. Data encryption is performed with a symmetric key in a multiplicative group, and hiding the encrypted key using a multiplying operation is more effective.

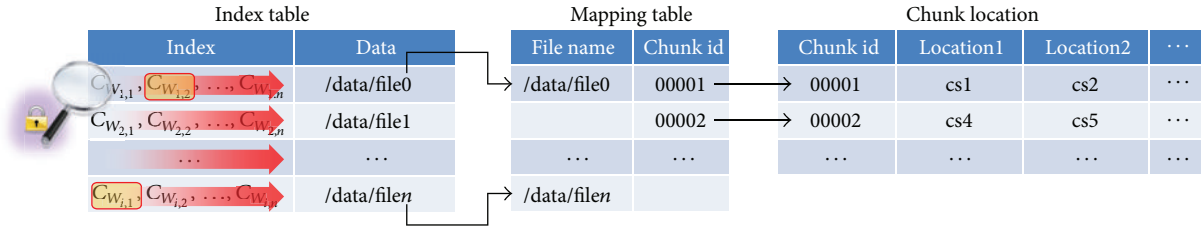

Figure 1: Existing index structure of PRES.

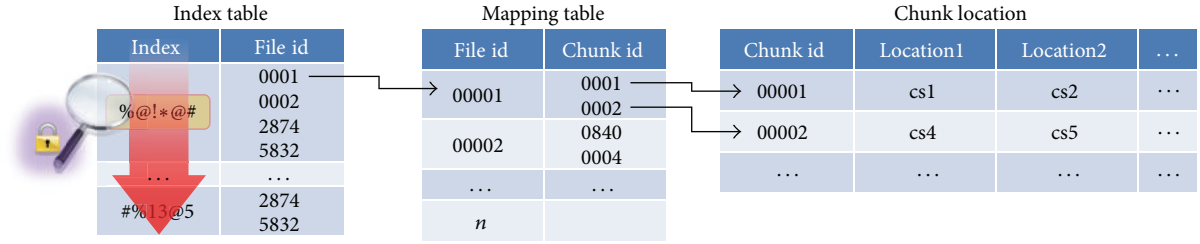

Figure 2: Proposed index structure of PRES.

(v) Storage volume efficiency: a variety of distributed file systems have been developed to provide cloud storage services. These systems store the index in master server's memory for faster data retrieving. In other words, the storage capacity of the index has limitations. Due to these circumstances, a service provider uses this technique to merge the repeated keyword and optimize the index. The server cannot merge duplicated keywords, in the case of existing schemes, using the same structure as in Figure 1. In this the structure the index capacity will also increase rapidly depending on the number of documents. However, if we adopt the structure shown in Figure 2, index capacity management will be more efficient.

(vi) Sharing efficiency among users: encrypted data must be retrieved from saved remote data and be securely and efficiently shared with those users who use an unreliable server. Cloud service providers should make shareable only the data that the data owner wishes to share with another user. The PRES papers most often propose previously used proxy reencryption (PRE). These schemes provide a once-only sharing function. In other words, B cannot share data with another user C with a similar scheme as the one used to share the data between users A and B. However, B is able to search and decrypt the shared data and then share it by saving it to the remote storage again through the PRES encryption process. The existing PRES is not sharing the shared data to B again, and additional decryption and encryption operations are needed to share the data again. Therefore, PRES needs to consider a re-share operation.

(vii) Prevention of a collusion attack: the administrator of the remote storage is treated as an untrusted object, and the administrator may obtain unauthorized access to the data through collusion. Therefore, PRES proposed in the future needs to be safe from collusion attack.

3. Proposed Scheme

In this paper, a practical proxy reencryption scheme with a keyword search capability is proposed considering the structural characteristics of an entrusted cloud storage center. This paper describes what steps should be taken in a secure data storage, searching, and sharing scenario (refer Figure 3).

3.1. Notation

(i) $\|$: Concatenation.
(ii) p: Prime number.
(iii) n: Number of data.
(iv) m: Number of keyword on data.
(v) G: Cyclic additive group of order p.
(vi) G_T: Cyclic multiplicative group of order p.
(vii) g: Generator of G.
(viii) e: Bilinear map, $G \times G \rightarrow G_T$.
(ix) sk_*: $*$'s private key in Z_p.
(x) pk_*: $*$'s public key in G.
(xi) pd_i: ith plain data.
(xii) ed_i: ith encrypted data.
(xiii) k_i: ith data encryption key ($i = 1 \sim n$).
(xiv) $w_{i,j}$: jth keyword on ith data ($j = 1 \sim m$).
(xv) $\text{Enc}_k(\,)$: Symmetric key encryption by key k.

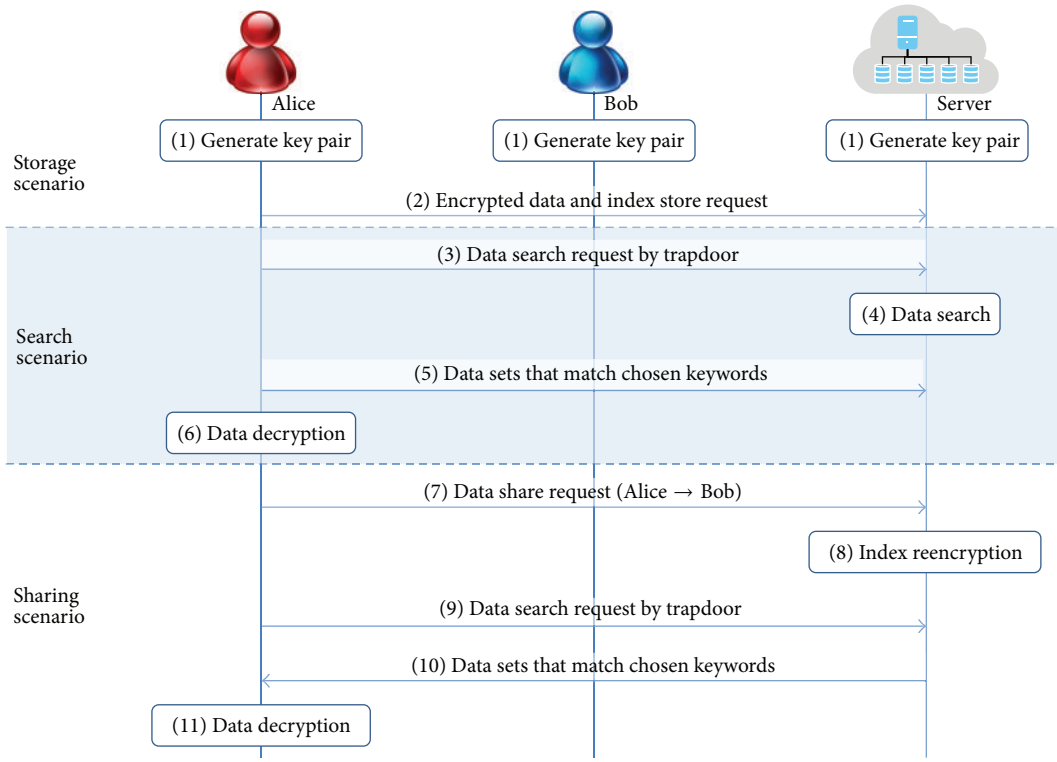

Figure 3: Flow chart of proposed scheme.

(xvi) $\mathrm{Dec}_k(\,)$: Symmetric key decryption by key k.

(xvii) W_i: Set of keyword on ith data $*$.

(xviii) $H_1(\,)$: Hash function, $\{0,1\}^* \rightarrow G$.

(xix) $H_2(\,)$: Hash function, $G \rightarrow G$.

(xx) T_*: Trapdoor searching keyword $*$.

3.2. Definition. The detailed steps performed by the proposed scheme are as follows.

(i) KeyGen: the users of the outsourced storage generate public key pairs prior to using the service. The storage outsourcing server should not store the user's private key. If the private key is leaked, an attacker can generate a trapdoor by acting as the owner of the private key. Thus, we generate a key pair based on the discrete logarithm problem (DLP).

(ii) $\mathrm{Enc}(\mathrm{sk}, W, \mathrm{pd}) \rightarrow E, \mathrm{ed}$: the data owner creates the encrypted index, E, and encrypted data, ed, which only the owner can search by inputting his or her own private key, sk, and a set of keywords, W, which are sent to the master server.

(iii) $\mathrm{TGen}(\mathrm{sk}, w) \rightarrow T_w$: to search the data safely, the user creates a trapdoor, T_w, which does not leak information related to the keyword w, which is being searched for using the private key sk. The trapdoor is sent to the master server. The storage outsourcing administrator should not be able to access information via a trapdoor.

(iv) $\mathrm{Test}(E, T_w) \rightarrow$ "yes" or "no": using the trapdoor generated by the user's private key and the search keyword, the server performs a test to confirm whether the encrypted data contain the keywords. If the cipher text contains the keyword specified, the server sends a "yes" to the user and a "no" if it does not. Thus, the server cannot learn anything about the keywords or the data.

(v) $\mathrm{RKGen}(\mathrm{sk}_a, h(\mathrm{sk}_b)) \rightarrow rk_{a\rightarrow b}$: the data owner A creates a reencryption key, $rk_{a\rightarrow b}$, to create a data index for sharing that B can search. The reencryption key is created with the data owner's secret key sk_a, and the hashed secret key $h(\mathrm{sk}_b)$ of the user who will be sharing the data.

(vi) $\mathrm{REnc}(\mathrm{sk}_a, \mathrm{pk}_b, E_a) \rightarrow E_b$: the data owner a creates a parameter to generate a data index for sharing that can be searched by b. This parameter is created using the data owner's private key sk_a and the public key pk_b of the user who will be sharing the data. The master server creates a new index, E_b, which b can use to search via the trapdoor.

(vii) $\mathrm{Dec}(\mathrm{sk}, E, \mathrm{ed}) \rightarrow \mathrm{pd}$: the rightful owner of the encrypted data uses their private key to decrypt the encrypted data.

3.3. Storage Scenario. The proposed scheme considers the outsourced storage structure so an encrypting index used for sharing and searching is stored on the master server. We assume that each user has received a key pair before using

the storage outsourcing service (refer to Step 1). The user encrypts the necessary keywords during data searching so they can perform their own search later and send this to the master server (refer to Step 2). The master server sends chunk information to the user for data storage, who then divides the data into chunks and stores it on the designated chunk server (see Figure 4).

Step 1 (key generation (KeyGen)). Each storage outsourcing service user generates a key pair:

$$x \in Z_p \text{ selection}$$
$$\text{sk} = x \text{ setting up}$$
$$\text{pk} = g^x \text{ setting up.}$$

Step 2 (index and data encryption (Enc)). The data owner generates an encrypted index which can be used for searching securely:

$$\text{Alice: } k_i \text{ selection}$$
$$ap = \text{pk}_a^{h(k_i)}$$
$$ew_{i,j} = e(\text{pk}_s, H_1(w_{i\cdot j}))^{h_2(h_2(\text{sk}_a)||w_{i,j})}$$
$$EW = \{ew_{i,1}, ew_{i,2}, \ldots, ew_{i,m}\}$$
$$EK_i = e(H_2(\text{pk}_s), g)^{hk} \cdot k_i \text{ output encrypted index for the master server}$$
$$ED_i = \text{Enc}_{k_i}(PD_i) \text{ output encrypted data for the chunk server}$$
$$A \rightarrow S\text{: } EW, EK_i, ED_i.$$

3.4. Search Scenario. The user sends a trapdoor that can search data without exposing keyword information to the master server (refer to Step 1). The master server searches for the data with the keyword in the encrypted index using the trapdoor and then sends the chunk information that corresponds to the data to the user (refer to Step 2). The retrieved data is decrypted by the legitimate user (refer to Step 3). The user acquires the data by summing each chunk received from the chunk server that stores the data (see Figure 5).

Step 1 (trapdoor generation (TGen)). A user, a, who wants to search the data generates a trapdoor using the keywords and his or her secret key:

$$\text{Alice} \rightarrow \text{Server: } T_w = H_1(w)^{-\text{sk}_a} \parallel h_2(h_2(\text{sk}_a) \parallel w).$$

Step 2 (Test). To confirm that the data contains the keywords sought by the user, the user performs the following tests with the public key, trapdoor, and crypt obtained from the server:

$$\begin{aligned} \text{Server: } ew &= ?e\left(\text{pk}_a^s, H_1(w)^{-\text{sk}_a}\right)^{h_2(h_2(\text{sk}_a)\|w)} \\ &= e\left(g^{a\cdot s}, H_1(w)^{-a}\right)^{h_2(h_2(\text{sk}_a)\|w)} \\ &= e\left(g^s, H_1(w)\right)^{h_2(h_2(\text{sk}_a)\|w)} \\ &= e\left(\text{pk}_s, H_1(w)\right)^{h_2(h_2(\text{sk}_a)\|w)} \end{aligned} \tag{9}$$

Step 3 (decryption (Dec)). The user can perform the following decryption using their private key and the crypt obtained from the server:

$$\begin{aligned} \text{Alice: } k_i &= EK/e\left(ap, H_2(\text{pk}_s)\right)^{-\text{sk}_a} \\ &= EK/e\left(\text{pk}_a^{h(k)}, H_2(\text{pk}_s)\right)^{-a} \\ &= EK/e\left(g^{h(k)}, H_2(\text{pk}_s)\right)\text{: output decryption key} \\ PD_i &= \text{Dec}_{k_i}(ED_i)\text{: output decrypted data.} \end{aligned} \tag{10}$$

3.5. Sharing Scenario. To share data with the desired user and to allow the shared users to share data freely with another user, reencryption needs to be performed to allow the shared users to search only the encrypted index. Many parameters are required to implement proxy reencryption and a separate searchable encryption scheme for secure data sharing in a storage outsourcing environment, which reduces the storage volume efficiency. Therefore, we propose an algorithm that provides both functions simultaneously. First, parameter A is generated to allow index sharing with another user, which is sent to the storage outsourcing provider by the owner of the data (refer to Step 1). Next, the storage outsourcing provider changes the owner's index with respect to the data sharing target. Shared (reencrypted) data searching is then possible, as shown in Steps 2–5. A user who acquires the data sharing index can always search for the corresponding data using keywords and then download it (see Figure 6).

Step 1 (reencryption key generation (RKGen)). If the data owner wants to share data with other users, he or she can generate keys for reencryption. If user a wants to share data with user b, a generates parameter A' using a's secret key and b's public key, as follows:

$$\text{Bob} \rightarrow \text{Alice: } h(b)$$
$$\text{Alice: } rk_{a\rightarrow b} = h_2(h_2(b), w_{i\cdot j})/h_2(h_2(a), w_{i\cdot j})$$
$$ap' = \text{pk}_b^{h(k)}$$
$$\text{Alice} \rightarrow \text{Server: } rk_{a\rightarrow b} \parallel ap'.$$

Step 2 (reencryption (REnc)). If user a wants to share data with user b, a generates parameter $ew'_{i,j}$ using a's secret key and b's hashed secret key, as follows:

$$\begin{aligned} \text{Server: } ew_{i,j}^{rk_{a\rightarrow b}} &= ew_{i,j}^{h_2(h_2(\text{sk}_b),w_{i\cdot j})/h_2(h_2(\text{sk}_a),w_{i\cdot j})} \\ &= e\left(\text{pk}_s, H_1\left(w_{i\cdot j}\right)\right)^{h_2(h_2(\text{sk}_b)\|w_{i,j})}. \end{aligned} \tag{11}$$

Step 3 (trapdoor generation (TGen)). User b who wants to search the data, generates a trapdoor using the keywords and his or her secret key:

$$\text{Bob: } T_w = H_1(w)^{-\text{sk}_b} \parallel h_2\left(h_2\left(\text{sk}_b\right) \parallel w\right). \tag{12}$$

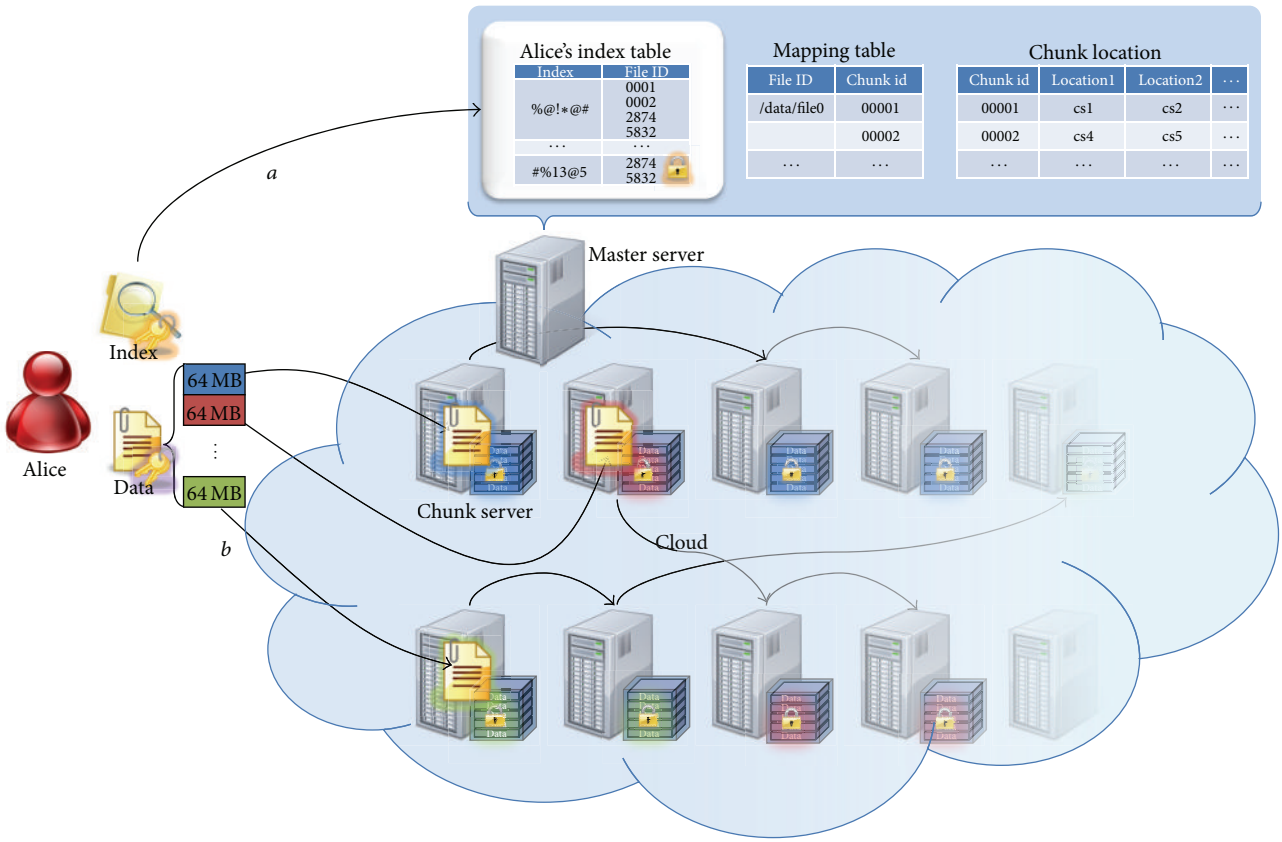

Figure 4: Storage scenario.

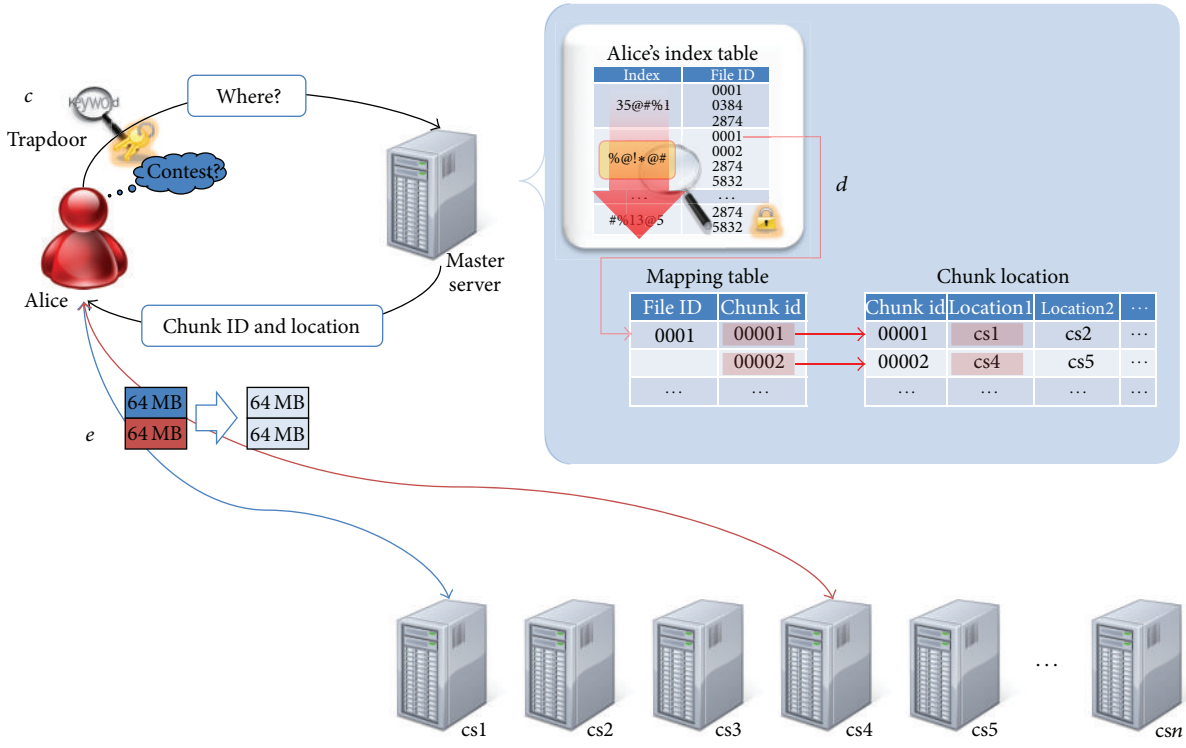

Figure 5: Search scenario.

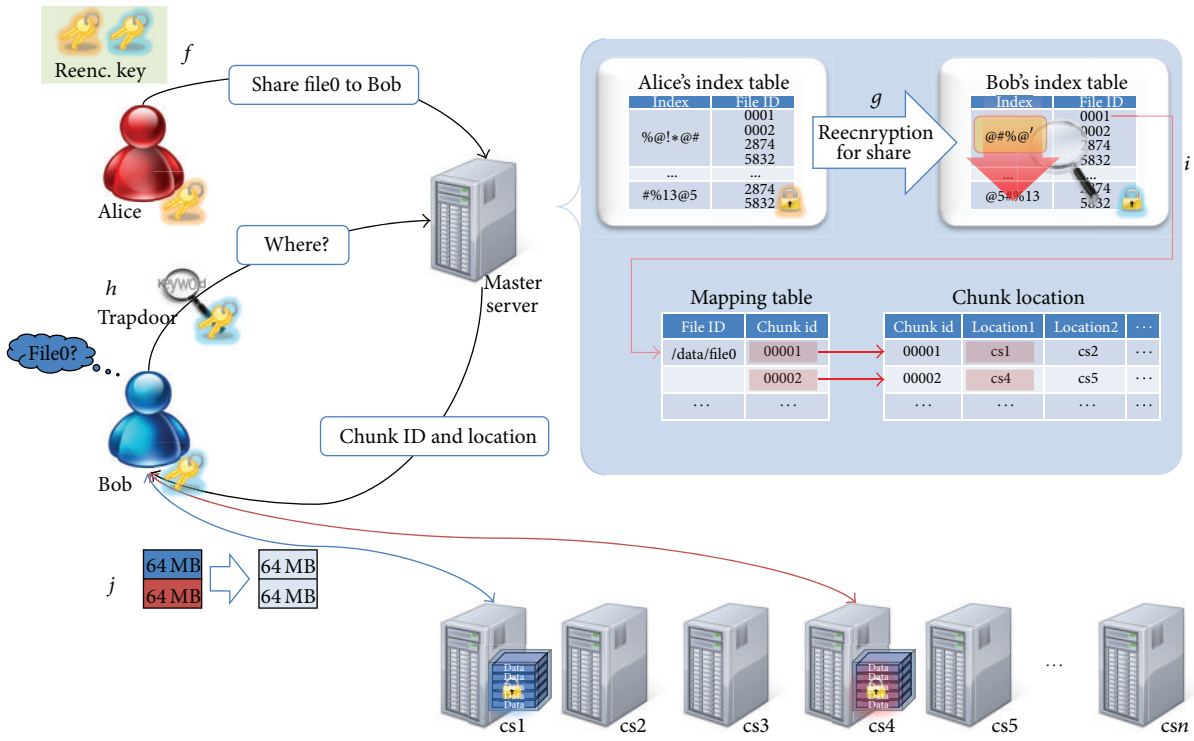

FIGURE 6: Sharing scenario.

Step 4 (test). To confirm that the data contains the keywords the user seeks, the server performs the following tests using bob's trapdoor. It checks the equality $ew =?e(\mathrm{pk}_b^s, H_1(w)^{-\mathrm{sk}_b})^{h_2(h_2(\mathrm{sk}_b)\|w)}$. If this is true, the output is "Yes" but "No" if not,

$$\begin{aligned}\text{Server: } ew &=?e\left(\mathrm{pk}_b^s, H_1(w)^{-\mathrm{sk}_b}\right)^{h_2(h_2(\mathrm{sk}_b)\|w)}\\ &= e\left(g^{b\cdot s}, H_1(w)^{-b}\right)^{h_2(h_2(\mathrm{sk}_b)\|w)}\\ &= e\left(g^s, H_1(w)\right)^{h_2(h_2(\mathrm{sk}_b)\|w)}\\ &= e\left(\mathrm{pk}_s, H_1(w)\right)^{h_2(h_2(\mathrm{sk}_b)\|w)}.\end{aligned} \tag{13}$$

Step 5 (decryption (Dec)). The user can perform the following decryption with his or her private key:

$$\begin{aligned}\text{Bob: } k_i &= EK/e\left(ap', H_2(\mathrm{pk}_s)\right)^{-\mathrm{sk}_b}\\ &= EK/e\left(\mathrm{pk}_b^{h(k)}, H_2(\mathrm{pk}_s)\right)^{-a}\\ &= EK/e\left(g^{h(k)}, H_2(\mathrm{pk}_s)\right)\text{: output decryption key}\\ PD_i &= \mathrm{Dec}_{k_i}(ED_i)\text{: output decrypted data.}\end{aligned} \tag{14}$$

4. Analysis

The proposed scheme satisfies the following requirements.

(i) Confidentiality: using pairing, the proposed scheme makes it difficult for a malicious third party to decrypt communication contents, even if they eavesdrop on communications between the client and the server.

(ii) Search speed: a quick index search is possible by using the index structure shown in Figure 2, and a user can check whether a document contains keywords by performing single pairing calculations, which increases the searching speed (refer Figure 7).

(iii) Traffic efficiency: keyword search and reencryption requires only one round of communication, so the scheme increases the communication volume efficiency.

(iv) Storage volume efficiency: to use a new index structure, the proposed scheme can reduce storage volume dramatically despite increasing the index document storage space compared to traditional schemes (refer Figure 8). Because, the proposed scheme can merge the same keywords.

(v) Calculation efficiency: the relatively simple pairing calculation implies that the proposed scheme allows users to generate indexes and search documents, as well as perform reencryption, which increases the calculation efficiency (refer Table 1).

(vi) Sharing efficiency among users: our scheme allows encrypted and stored data on an unreliable remote outsourced storage server to be shared safely and efficiently. In addition, our proposed scheme is different from existing schemes because it does not require the

Table 1: Calculation efficiency analysis.

	Chen's scheme					Proposed scheme				
	Exponential operation	Pairing operation	Hash operation	Multiply operation	Comparison operation	Exponential operation	Pairing operation	Hash operation	Multiply operation	Comparison operation
Kgen	u					u				
Enc	$2m + 4$	$m + 2$	$2m + 2$	1		$m + 1$	$m + 1$	$3m + 1$	1	
RKGen	2					2				
ReEnc	2	2	2				1			
TGen	1		1			1		3		
Test		1	1		$n * m$	1	1			c
Dec	1			1		1	1		1	

c: number of comparison operation on existing search scheme ($c \leq \log_2(n * m)$), m: number of keyword on document, n: number of all documents on cloud storage, u: number of user.

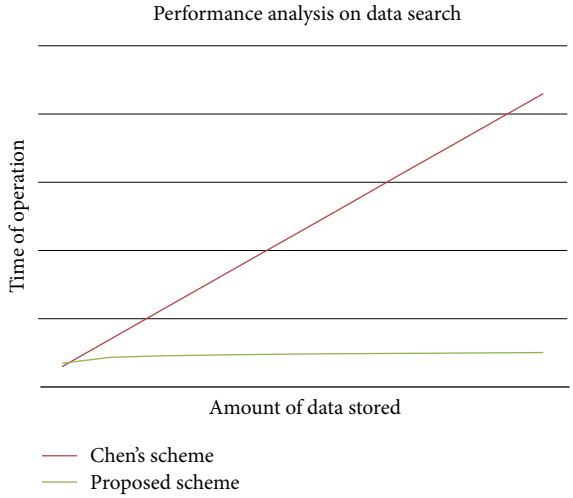

Figure 7: Search speed.

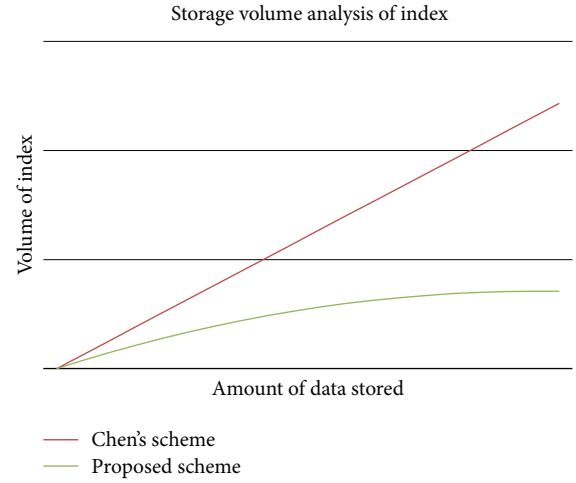

Figure 8: Storage volume.

shared subjects to be specified in advance, and no additional devices are required to manage the subjects who receive the shared data. Finally, if users want to re-share the data shared by the owner with other users, they only require one pairing calculation in an unreliable storage outsourcing environment.

(vii) Prevention of collusion attack: in the proposed scheme, each data set is encrypted by a different random key (for symmetric encryption). Therefore the sharing phase can be operated by only the lawful data owner. An unethical administrator cannot use a collusion attack, because the key is known only to the lawful data owner.

5. Conclusion

The advent of storage outsourcing services has allowed many users to store and access data. Recent studies of the application of searchable encryption technologies to storage outsourcing have attempted to ensure the security of data. However, most available searchable encryption technologies are inefficient when adding data sharing objects because they are based on e-mail environments, which determine the objects with which data can be shared. In a storage outsourcing environment, users upload data on their own and share the data in a safe manner. Therefore, the indexes and data are separated so available technologies are compatible with data storage outsourcing systems. After considering the requirements of the data storage outsourcing environment, we specified the security requirements and proposed a scheme that provides both functions simultaneously: a proxy reencryption function and a searchable encryption function. The proposed scheme provides a free sharing feature which has the more calculation efficiency than existing schemes. And we adopted the new index structure for fast searching data on cloud storage. It appears that search schemes based on multiple keywords will become important for ensuring flexibility and for facilitating searches during data storage outsourcing. In the future, it will be necessary to develop a reencryption system where an index containing multiple keywords of variable lengths can be encrypted and searched flexibly.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the MKE (The Ministry of Knowledge Economy), Republic of Korea, under the Information Technology Research Center (ITRC) support program (NIPA-2013-H0301-13-1003) supervised by the National IT Industry Promotion Agency (NIPA). This work was supported by the Soonchunhyang University Research Fund.

References

[1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 44–55, Berkeley, Calif, USA, May 2000.

[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," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 79–88, Alexandria, Va, USA, November 2006.

[4] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, 2004.

[5] D. Boneh and B. Waters, "Conjunctive, subset and range queries on encrypted data," in *Proceedings of the 4th Theory of Cryptography Conference*, Amsterdam, The Netherlands, 2007.

[6] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proceedings of the 1st International Conference on Pairing-Based Cryptography*, Tokyo, Japan, 2007.

[7] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proceedings of the 4th International Conference on Information Security Practice and Experience*, Sydney, Australia, 2008.

[8] S. Kamara and K. Lauter, "Cryptographic outsourcing storage," in *Proceedings of Workshops on Financial Cryptography and Data Security*, pp. 25–28, Canary Islands, Spain, 2010.

[9] M. Ion, G. Russello, and B. Crispo, "Enforcing multi-user access policies to encrypted cloud databases," in *Proceedings of the IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY '11)*, pp. 175–177, 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, pp. 262–267, 2011.

[11] Y. Yang, "Towards multi-user private keyword search for cloud computing," in *Proceedings of the IEEE 4th International Conference on Cloud Computing (CLOUD '11)*, pp. 758–759, Singapore, July 2011.

[12] J. Shao, Z. Cao, X. Liang, and H. Lin, "Proxy re-encryption with keyword search," *Information Sciences*, vol. 180, no. 13, pp. 2576–2587, 2010.

[13] X. Chen and Y. Li, "Efficient proxy re-encryption with private keyword searching in untrusted storage," *International Journal of Computer Network and Information Security*, vol. 3, no. 2, 2011.

[14] X. A. Wang, X. Huang, X. Yang, L. Liu, and X. Wu, "Further observation on proxy re-encryption with keyword search," *Journal of Systems and Software*, vol. 85, no. 3, pp. 643–654, 2012.