EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

PKIS: practical keyword index search on cloud datacenter

Hyun-A Park[1], Jae Hyun Park[2] and Dong Hoon Lee[1*]

Abstract

This paper highlights the importance of the interoperability of the encrypted DB in terms of the characteristics of DB and efficient schemes. Although most prior researches have developed efficient algorithms under the provable security, they do not focus on the interoperability of the encrypted DB. In order to address this lack of practical aspects, we conduct two practical approaches–efficiency and group search in cloud datacenter. The process of this paper is as follows: first, we create two schemes of efficiency and group search–practical keyword index search–I and II; second, we define and analyze group search secrecy and keyword index search privacy in our schemes; third, we experiment on efficient performances over our proposed encrypted DB. As the result, we summarize two major results: (1)our proposed schemes can support a secure group search without re-encrypting all documents under the group-key update and (2)our experiments represent that our scheme is approximately 935 times faster than Golle's scheme and about 16 times faster than Song's scheme for 10,000 documents. Based on our experiments and results, this paper has the following contributions: (1) in the current cloud computing environments, our schemes provide practical, realistic, and secure solutions over the encrypted DB and (2) this paper identifies the importance of interoperability with database management system for designing efficient schemes.

Keywords: keyword index search, encrypted document, group setting, DBMS, index list table, normalization, primary key, foreign key, group search secrecy, keyword index search privacy, cloud datacenter

1 Introduction

Cloud computing technologies have become a central issue in order to open a new digitalized information society by heterogeneous services and convergence of technologies. In the era of cloud computing, personal computer and storage have changed their functions and features in socio-technical perspectives: the functions of personal computers have changed their concerns from individual to centralized managerial ones; the features of storage have also transformed its boundaries from personal databases or Enterprise Resource Planning (ERP) severs to the datacenter in social storage systems [1,2].

In the cloud computing era, security research also encounters a variety of challenges and issues. Because the datacenter is made up of complex private information, and the datacenter is faced with the risks of information leakages and intruders or insiders' attacks. With these reasons, prior researchers have considered encryption as the most substantial way for protecting sensitive information as the last line of database defense.

1.1 Problem identification

In DB encryption, previous researchers have conducted the keyword index search over encrypted documents with various scenarios; however, the keyword index search scheme is inefficient and impractical aspects in a real world. The keyword index search enables a legitimate queries to search the encrypted documents with an encrypted keyword over the encrypted indexes without revealing any information on the query and documents, even to the server.

In most prior research, we find that the indexes of each data are stored by a row, not by a field (column) as another inefficient respect. The keyword index search schemes require at least a verifying test for every row of each data, so that the computational complexity of the

* Correspondence: donghlee@korea.ac.kr
[1]Graduate School of Information and Security, Korea University, 5-Ka, Anam-dong, Sungbuk-ku, Seoul 136-701, Korea
Full list of author information is available at the end of the article

previous schemes requires at least $O(n)$ if the total number of stored data is n. The computation or scanning over many fields within one row is not fast, while the computation or scanning within one field is relatively faster than in one row. Moreover, encryption algorithm needs many random factors, which makes it hard to apply efficient DB schema[a] to encrypted databases.

Our schemes are in the line of the keyword index search area, and this paper focuses on more practical approaches over the encrypted database to resolve the problems–the efficiency and group search of the encrypted database in the cloud datacenter service.

In this paper, we extend the search scope from between a server and a single user to the search between a server and group members (multiple users) in the cloud datacenter services, because current changing cloud computing technologies call for a variety of collaborations and cooperation among users in a certain social networking environment. These changing social networking environments require multiple users' information sharing in a certain organization; therefore, we propose the group key search of database encryption, when a group member shares his or her sensitive information among multiple users. Especially, sharing sensitive information should be encrypted by a group key in group search of database encryption. On the other hand, a group key has some problems to be used as a search key, because the group key has a dynamic property, i.e., a person may join or leave from the group. When a member leaves from a group, all data accessible to the group should not be accessible any more. It could be resolved by updating a group key, and the leaving member must not compute a new group key. On the other hand, when a member joins a group, he or she should obtain all of the previous group keys in order to access all of the group data. This problem, a member joins a group, makes design much harder. A naive solution is to decrypt all documents of the group and re-encrypt the documents by the new group key according to every membership change. Yet this solution entails a large amount of computational overheads.

In prior research, most schemes have not considered practical usages, while [3,4] worked on the search schemes of dynamic group membership changes without re-encrypting documents. Park et al.'s scheme [3] is relatively faster than that of Wang et al. [4]. Wang et al.'s is based on bilinear, while Park et al. utilized the reversed hash key chains and bloom filters. The faster Park et al.'s scheme has a potential problem related to 'group member leave'. This paper, therefore, seeks to fix this proposed problem from Park et al.'s scheme–the reversed hash key chains, and it also develops novel efficient schemes with the experiments.

1.2 Key idea and contribution

The previous schemes have focused on the development of new encryption algorithms, while we apply general DB schema to the encrypted database instead of developing an efficient encryption algorithm. Based on this key idea, we devise two tables and store all indexes for all documents in one field (column). The two tables enable to build database normalization[b] by applying primary keys and foreign keys into the tables. These properties of two tables enable the server to directly access the data that a user wants to search without any verification processes for every row.

Based on these two tables for efficiency, we construct PKIS-I with the reversed one-way hash key chain and PKIS-II with the key matching table, for the group search.

Through PKIS-I and PKIS-II, we summarize the results as follows:

1) Efficiency

- Compared to computational complexity during the search process, our schemes' is $O(1)$, while other previous papers' is at least $O(n)$.
- Our experiments represent our scheme is approximately 935 times faster than Golle's scheme and about 16 times faster than Song's scheme for 10,000 documents.

2) Group search

- By re-encrypting keywords or documents with the group manager (GM)'s secret key k_c, we resolved the encrypted database group search problem in cloud service.
- Whenever every membership change, our schemes can support a secure group search without re-encrypting all documents.

3) Security

- We made definitions on group search secrecy and keyword index search privacy and analyzed them.

Therefore, this paper has two contributions as follows: (1) our schemes provide practical and realistic encrypted DB solutions in the cloud computing environments and (2) this paper identifies the importance of interoperability with DBMS as well as developing algorithms, to design efficient schemes.

1.3 Related works

The search systems research of encrypted data has been regarded as an active area with various scenarios. In this section, we review the prior papers in search systems on encrypted database.

Song et al. [5] firstly proposed a sequential scanning search algorithm, searchable symmetric key encryption, over entire documents by using stream and block ciphers. Following this idea, most researches have been conducted on the keyword index search. Boneh et al. [6] proposed a keyword search with a public key system, where they defined the concept of a public key encryption with keyword search (PEKS) and showed that PEKS implies identity-based encryption; however, the converse is currently an open problem. Chang et al. [7] suggested two index search schemes with the idea of pre-built dictionaries. Goh [8] formulated a security model for indexes known as semantic security (or indistinguishability) against an adaptive chosen keyword attack (IND-CKA), and they also proposed an secure index scheme in the model. Waters et al. [9] published the building of an encrypted and a searchable audit log, which searches the encrypted log with extracted keywords. Byun et al. [10] raised a serious vulnerability of public key-based keyword search schemes, which are susceptible to an off-line keyword guessing attack through much smaller space than passwords.

In addition, some proposed schemes extend the types of encrypted data queries. Boneh and Waters [11] suggested a public key system in order to support queries for testing any predicate on encrypted data with tokens produced by a secret key. They constructed comparison systems, subset queries, and conjunctive versions of these predicates, which introduce a primitive, hidden vector encryption. Hacigum*üs* et al. [12] proposed the method of range queries on encrypted data in the Database As a Service (DAS) model by using privacy homomorphism that allows basic arithmetic (+, -, ×) on encrypted data. Golle et al. [13] firstly proposed an efficient conjunctive keyword search over encrypted data and their scheme constructs a keyword field.

Hwang et al. [14] constructed a conjunctive keyword search scheme for group users, based on the public key. Wang et al. [4] developed threshold privacy preserving keyword search scheme. These schemes cannot support dynamic groups, while Park et al. [3] firstly proposed search schemes of dynamic groups, and their search schemes deal with membership changes without re-encrypting documents for each change of membership. Later, Wang et al. [15] built conjunctive keyword searches on encrypted data without keyword fields, and they applied these searches to the setting of dynamic groups.

Zerr et al. [16] worked on the problem of supporting keyword search for sensitive unstructured documents shared within collaboration groups. They proposed r-confidential Zerber indexing facility for sensitive documents, and they utilized secret splitting and term merging to provide tunable limits on information leakage, even under statistical attacks. As they admitted, this proposed indexing scheme would be unattainable in practice, and their scheme is inefficient. In succession, Zerr et al. [17] published Top-K retrieval algorithm from $ZERBER^{+R}$. In this work, they focused on ranked keyword search, term frequencies, and a novel relevance score transformation function. Here, the function in novel relevance score transformation hides the term-specific distribution of relevance score values, and it makes the scores of different terms indistinguishable. The authors of [18,19] also handled with the same problems.

Wang et al. [20] considered the problem, concerning effective yet secure ranked keyword search over encrypted cloud data. In order to achieve practical performance, Wang et al. proposed a definition for ranked searchable symmetric encryption and used order-preserving symmetric encryption. Yet [20] is not a design for the group search. Cao et al. firstly explored the problem of multi-keyword ranked search over encrypted cloud data (MRSE), and they established a set of strict privacy requirements for such a secure cloud data utilization system to become a reality [21]. They proposed a basic MRSE scheme using secure inner product and then improved this scheme in order to meet different privacy requirements in two levels of threat models. Additionally, Zerr et al.'s schemes are not Boolean operation on multiple keywords searches in traditional searchable encryption schemes but they are ranked search operation. The evaluation methods and security requirements such as term frequency[c] are different. Hence, the comparisons with our schemes are actually meaningless.

As for the papers about encrypted data in cloud computing, additionally, there are Li et al.'s [22] and Yu et al.'s [23]. Li et al. handled with the problem of authorized private keyword searches (APKS) over encrypted data in cloud computing, where multiple data owners encrypt their records along with a keyword index to allow searches by multiple users. Their two novel solutions for APKS are based on hierarchical predicate encryption, which uses pairing-based cryptography. Yu et al. proposed a secure and scalable fine-grained data access control scheme for cloud computing. In order to achieve this goal, they combined the techniques of attribute-based encryption, proxy re-encryption, and lazy re-encryption, which are also pairing-based cryptography.

2 Preliminaries

2.1 Keyword index search scheme

In general, keyword index search schemes consist of setup and searching processes. In the setup process, a client uploads encrypted data together with its indexes (also called searchable information) on a database server, and the indexes are encrypted keywords for searching the data. To search data with a keyword in the searching process, a user generates a trapdoor and sends it to the server. Here, the trapdoor is the encryption of the keyword and provides only search capabilities to the server without revealing any information about the keyword. The database manager runs the test algorithm with the indexes and the trapdoor as input to find the corresponding data. That is, this searching verification is performed on the indexes rather than on the encrypted data. The results are returned to the client, and the client finally decrypts the results and sends them back to the user.

2.2 System environments

2.2.1 Multiple user setting

Our system is devised for a certain group organization, which includes many departments such as government offices, organizations, or enterprises. This group includes subgroups (g_1, g_2, ..., g_7) and their members (p_1, p_2, ..., p_{15}). This paper identifies a group as a set of people with the same aims, and the group organizes the people working together. In this paper, we focus on a group search, because private search is possible through the same process as well.

2.2.2 Cloud datacenter service and modified DAS model

Our application storage system is a datacenter for the cloud storage service.[d] The users of group members store their sharing documents in a datacenter, not their own server. In this case, we cannot guarantee that the datacenter server managers are trust; therefore, we utilize the cryptographic method for the data. This is similar to DAS model of [12]. In the DAS model, a client is trustworthy, while users' data are stored in and managed by an untrustworthy server. A client has a restricted computational power and storage and relies on the server for a mass computational power and storage. A server can be an inside attacker and is not allowed to read the data. Hence, the encryption key should not be known to the server (or the database administrator). Data privacy is assured under the conditions that a client does not share encryption keys, metadata or original data with any party.

Here, we modify the DAS model into our application system. Our scheme is made up of three parties: (1) users of group members, (2) a group manager GM, and (3) a datacenter server DS.

Users of group members are the owners of documents, and they are registered in their organization. **GM** plays a similar role of a client server, and it is a trusted party in our scheme. In our scheme, the GM manages the group session keys and the search keys of all groups, for secure communication and secure keyword index search.

DS is not a trustable party in our scheme. Hence, all of the documents in a server should be encrypted and querying keywords should be also encrypted. One of the most important things is that there is no decryption by a server through all processes.

2.3 Notations

- TG: a huge hierarchical group
- g_i: ith small group of G
- g_i^j: a small group g_i at jth session
- D_n: nth documents
- W_n: keywords list of D_n
- w_n^i: ith keyword of W_n
- d_n: identifier of D_n
- gk_i: group session key of a small group g_i
- ik_i: index generation key of a small group g_i
- dk_i: documents encryption key of a small group g_i
- gk_i^j: group session key of g_i at jth session
- ik_i^j: index generation key of g_i at jth session
- dk_i^j: documents encryption key of g_i at jth session
- k_c: GM's secret key
- $f(\cdot)$: pseudorandom function (PRF)
- $h(\cdot)$: one-way hash function

2.4 Definitions

Definition 1. One-Way Hash Key Chain

It is generated by selecting the last value at random and applying a one-way hash function h repeatedly. Note that the initially chosen value is the last value of the key chain. The followings are two properties of a one-way hash chain [24].

- **Property 1** : Anybody can deduce that an earlier value k_i belongs to the one-way key chain by using the later value k_j of the chain and by checking $h^{j-i}(k_j)$ which equals k_i with the later value k_j.
- **Property 2** : Given the latest released value k_i of a one-way key chain, an adversary cannot find a later value k_j such that $h^{j-i}(k_j)$ equals k_i. Even when value k_{i+1} is released, the second pre-image collision resistant property prevents an adversary from finding k'_{i+1} different from k_{i+1} such that $h(k_{i+1})$ equals k_i.

Definition 2. PRF We say that '$F : K_f \times X \rightarrow Y$ is (t, q, e)-secure PRF' if every oracle algorithm A making at most q oracle queries and with running time at most t has advantage $Adv_A < e$. The advantage is defined as $Adv_A = |Pr[A^{F_k} = 1] - Pr[A^R = 1]|$ where R represents a random function selected uniformly from the set of all maps from X to Y, in which the probabilities are taken over the choice of k and R [5].

2.5 Algorithm

- **SysPara**(1^k). It takes an input as a security parameter k and outputs a system parameter λ. λ determines elements in order to set the encrypted database system such as the size of database, encryption/decryption algorithm, functions, the size of parameters, and so on.
- **KeyGen**(λ). Taking λ as an input, this algorithm generates users' group session key set $\{g_k\}$, index generation key set $\{ik\}$, and document encryption key set $\{dk\}$.
- **IndGen**(ik, W). Inputs of algorithm *IndGen* are an index generation key ik and a keyword set W. Output is index list table.
- **DocEnc**(dk, D). Given a document encryption key dk and a document D, this algorithm outputs an encrypted document.
- **TrapGen**(w, ik). This algorithm takes a keyword w and index generation key ik. It encrypts the keyword w with index generation key ik and returns the encryption value, which is the trapdoor T_w for the keyword w.
- **Retrieval**(T_w). This algorithm takes input as trapdoor T_w. If there exist matching values to the trapdoor T_w in an index list, then it outputs the encrypted documents that are mapped to the identifiers of the matching values in the index list table.
- **Dec**($E(D)$, dk). Given a document encryption key dk and encrypted document $E(D)$, it outputs a plaintext document D.

3 Construction Of Practical Keyword Index Search-I (PKIS-I)

Our scheme PKIS largely comprises of two parts; (1) uploading phase and (2) downloading phase. The uploading phase consists of four algorithms of *SysPara*; *KeyGen*; *IndGen*; *DocEnc*. The downloading phase is composed of three algorithms of *TrapGen*; *Retrieval*; *Dec*.

PKIS-I's group key generation method is based on [3]. However, in [3], SIS-G has a big potential problem. If one of group members would reveal his/her group key to a server, the server could know all of the previous documents of the group members. In order to resolve this problem, we add a re-encryption process through GM and propose a new practical scheme with normalized database tables over encrypted documents in a keyword index search protocol area.

3.1 Uploading phase

3.1.1 SysPara(1^k) construction

With the algorithm *SysPara*(1^k), GM generates system parameter $\lambda = (f(\cdot), h(\cdot), q)$. $f: \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$ is a PRF and $h: \{0,1\}^* \to \{0,1\}^k$ is one-way hash function. q is the length of one-way hash key chain.

3.1.2 KeyGen(λ) construction

In this construction, group search keys are generated. With system parameter λ, GM generates group session keys $\{gk_i^j\}$, index generation keys $\{ik_i^j\}$, and document encryption keys $\{dk_i^j\}$, where index generation keys and document encryption keys are called as search keys. The search keys are reversely generated by one-way hash key chains. At first, the last key of a key chain is selected (i.e. ik_1^q and dk_1^q, if the length of a key chain is q). GM applies the last key to a hash function repeatedly and computes all other keys until the first key comes out. It can be expressed like this: $ik_1^i = h(ik_1^{i+1})$, $dk_1^i = h(dk_1^{i+1})$ where $i \in [1, q-1]$. In more detail;

$$
\begin{aligned}
&\{ik_1^i\} = \{ik_1^q \in_R \{0,1\}^k,\\
&h(ik_1^q) = ik_1^{q-1},\\
&h(ik_1^{q-1}) = ik_1^{q-2},\\
&\cdots\cdots\\
&h(ik_1^4) = ik_1^3,\\
&h(ik_1^3) = ik_1^2,\\
&h(ik_1^2) = ik_1^1\}.
\end{aligned}
$$

$$
\begin{aligned}
&\{dk_1^i\} = \{dk_1^q \in R\{0,1\}^k,\\
&h(dk_1^q) = dk_1^{q-1},\\
&h(dk_1^{q-1}) = dk_1^{q-2},\\
&\cdots\cdots\\
&h(dk_1^4) = dk_1^3,\\
&h(dk_1^3) = dk_1^2,\\
&h(dk_1^2) = dk_1^1\}.
\end{aligned}
$$

For example, if an event of a session-change happens for a subgroup g_1, the first session is changed into the second session and then the group session key, a document encryption key, and an index generation key are changed like this: $gk_1^1 \to gk_1^2$, $dk_1^1 \to dk_1^2$, $ik_1^1 \to ik_1^2$.

One-way hash function h plays the important role of group search key in PKIS-I. One-wayness property of hash function can prohibit a leaving member from computing new keys after leaving the group. But any newly joining member can obtain all previous keys through applying the current key to hash function h repeatedly.

This eliminates decryption and re-encryption of the previous documents.

These search keys are distributed to all of the group members every membership change. For example, in the second session, a member of subgroup g_1 receives a new group session key gk_1^2 at first. This group session key can be distributed by GM with well-known group key protocols, such as one in [25]. Then, dk_1^2 and ik_1^2, which are computed in advance by the hash key chain, are encrypted with gk_1^2 and transferred to all members of subgroup g_1. It is illustrated in Figure 1.

3.1.3 IndGen(ik, W) and DocEnc(dk, D) construction

When a user stores documents D_n and its keywords $W_n = \{w_{n,1}, w_{n,2},...\}$ in a server, he encrypts the document and keywords with the algorithms *DocEnc* and *IndGen*. For a member of a small group g_i in the jth session, the encrypted document and indexes are generated as follows;

$$\{d_n, f_{dk_i^j}(D_n), f_{ik_i^j}(w_{n,1}), f_{ik_i^j}(w_{n,2}), \ldots\}$$

$f_{ik_i^j}(w_{n,1}), f_{ik_i^j}(w_{n,2}), \cdots$ are indexes that are the encrypted keywords. The user sends the encrypted document and indexes to GM.

3.1.4 Database update

Receiving the encrypted document and its indexes, GM re-encrypts them with his security key k_c. After this, GM sends them to a datacenter server DS. DS adds the received data to the tables of 'Index List' and 'Encrypted Document' every uploading time. 'Index List' is composed of indexes and their document identifiers as follows: $f_{k_c}(f_{ik_i^j}(w_{n,1})),\ f_{k_c}(d_n);\ f_{k_c}(f_{ik_i^j}(w_{n,2})),$ $f_{k_c}(d_n),\ f_{k_c}(d_n)$. Table 1 shows some parts of index list table. Then, DS stores an identifier $f_{k_c}(d_n)$ and encrypted documents $f_{k_c}(f_{dk_1^2}(D_n))$ in a row like Table 2. Namely, PKIS is composed of two tables, where $f_{k_c}(d_n)$ plays a role of a pointer as well as an identifier of D_n.

Since an index list is made by this way, we can make a relational DB by applying primary key and foreign key into PKIS. The 'Index' and 'Identifier of Document' of Table 1 are defined as 'primary key', and 'Identifier of Document' of Table 2 is defined as 'foreign key'. There is no computation to test and to search in a datacenter server. We can diminish the gap from general plaintext search systems through minimizing computational overhead in the retrieval stage and applying efficient DB schema.

3.2 Downloading phase

3.2.1 TrapGen(w, ik) construction

Algorithm *TrapGen*(w, ik) outputs trapdoors for a keyword w. We assume again that the user of group g_1 at the second session wants to search a keyword w. The keyword w may be included in the document at the second session or/and the first session. Therefore, the user has to generate two trapdoors encrypted with ik_1^1 and ik_1^2. That is, a user has to generate the trapdoors as many as the number of session-changes, which is possible because a user can compute all the previous search keys by applying the current search key to hash function h repeatedly. Then, the user computes trapdoors using the same method as index generation and sends them to GM. GM re-encrypts them with his secret key and then queries a datacenter server DS with the trapdoors. For a member of a small group g_i in the jth session, the trapdoors for a keyword w are as follows;

$$\begin{aligned} T_w &= \{f_{k_c}(f_{ik_i^s}(w)),\ 1 \le s \le j\} \\ &= \{f_{k_c}(f_{ik_i^1}(w)),\ f_{k_c}(f_{ik_i^2}(w)),\ \ldots, f_{k_c}(f_{ik_i^j}(w))\} \end{aligned}$$

3.2.2 Retrieval(T_w) and Dec(E(D), dk) construction

By the algorithm *Retrieval*, at first, DS searches the same values as the querying trapdoors in the 'Index' field of Table 1 and finds out the matching values to 'Index' and 'Identifier of Document'. Then, DS searches the same values as 'Identifier of Document' in Table 2 and returns the matching 'Encrypted Document's to GM. GM decrypts them with his secure key k_c and sends them to the user again. The user decrypts them with his/her group document encryption key.

Figure 1 describes the whole process of PKIS-I.

4 Construction Of Practical Keyword Index Search–II (PKIS-II)

In PKIS-II, the main difference from PKIS-I is that the search keys are not changed but fixed, irrespectively of membership changes. GM keeps the key matching information for groups, which consists of all of the group session keys and group search keys for each group. All users of group members do not know their group search keys. The only thing they know is a group session key. Instead, GM takes users' places for search processes.

The operative processes are similar to PKIS-I.

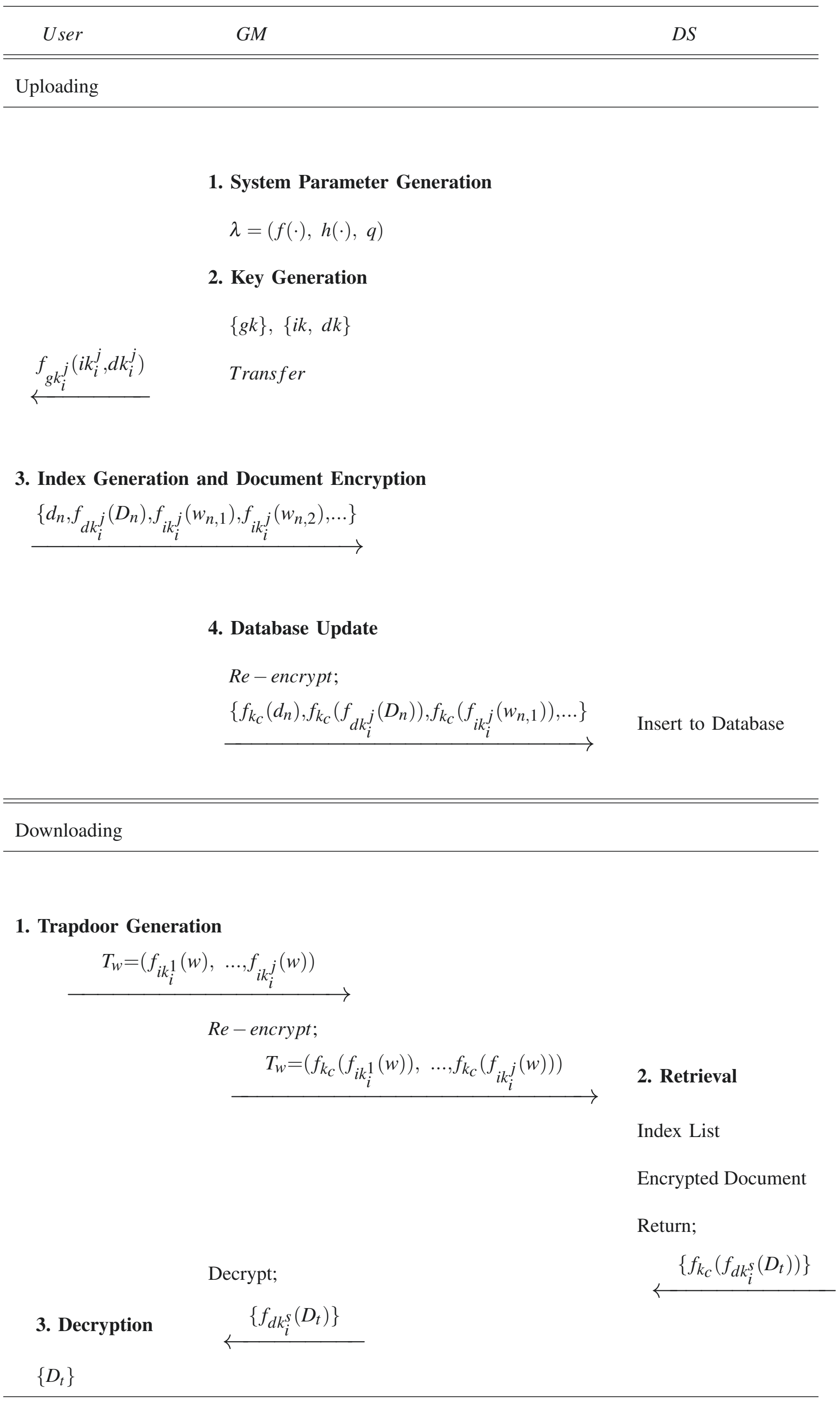
4.1 Uploading phase

4.1.1 SysPara(1^k) construction

This process is the same as PKIS-I.

4.1.2 KeyGen(λ) construction

GM generates group session keys, index generation keys, and document encryption keys for each group and stores them in a key matching table. In PKIS-II, if a session-change happens, for example of a subgroup g_1 from the first session to the second session, then the group session key is changed from gk_1^1 to gk_1^2. However, the search keys of document encryption key dk_1 and index encryption key ik_1 are unchanged and remain still as dk_1



User | *GM* | *DS*

Uploading

1. System Parameter Generation

$\lambda = (f(\cdot),\ h(\cdot),\ q)$

2. Key Generation

$\{gk\},\ \{ik,\ dk\}$

Transfer

$\xleftarrow{f_{gk_i^j}(ik_i^j, dk_i^j)}$

3. Index Generation and Document Encryption

$\xrightarrow{\{d_n, f_{dk_i^j}(D_n), f_{ik_i^j}(w_{n,1}), f_{ik_i^j}(w_{n,2}), \ldots\}}$

4. Database Update

$Re-encrypt;$

$\xrightarrow{\{f_{k_C}(d_n), f_{k_C}(f_{dk_i^j}(D_n)), f_{k_C}(f_{ik_i^j}(w_{n,1})), \ldots\}}$ Insert to Database

Downloading

1. Trapdoor Generation

$\xrightarrow{T_w = (f_{ik_i^1}(w),\ \ldots, f_{ik_i^j}(w))}$

$Re-encrypt;$

$\xrightarrow{T_w = (f_{k_C}(f_{ik_i^1}(w)),\ \ldots, f_{k_C}(f_{ik_i^j}(w)))}$

2. Retrieval

Index List

Encrypted Document

Return;

$\xleftarrow{\{f_{k_C}(f_{dk_i^s}(D_t))\}}$

Decrypt;

3. Decryption

$\xleftarrow{\{f_{dk_i^s}(D_t)\}}$

$\{D_t\}$

Figure 1 The whole process of PKIS-I.

Table 1 Index list

Index	Identifier of document
$f_{k_c}(f_{ik_1^1}(w_{n,1}))$	$f_{k_c}(d_1)$
$f_{k_c}(f_{ik_1^1}(w_{1,2}))$	$f_{k_c}(d_1)$
...	...
$f_{k_c}(f_{ik_1^1}(w_{1,t}))$	$f_{k_c}(d_1)$
$f_{k_c}(f_{ik_1^2}(w_{2,1}))$	$f_{k_c}(d_2)$
$f_{k_c}(f_{ik_1^2}(w_{2,2}))$	$f_{k_c}(d_2)$
...	...
$f_{k_c}(f_{ik_1^2}(w_{2,t}))$	$f_{k_c}(d_2)$
...	...
$f_{k_c}(f_{ik_{11}^{13}}(w_{114,1}))$	$f_{k_c}(d_{114})$
...	...
$f_{k_c}(f_{ik_{11}^{13}}(w_{114,t}))$	$f_{k_c}(d_{114})$
...	...
$f_{k_c}(f_{ik_i^s}(w_{n,t}))$	$f_{k_c}(d_n)$

and ik_1. When needed, they can be encrypted with GM's secret key k_c.

4.1.3 IndGen(ik, W) and DocEnc(dk, D) construction

When a user stores a document D_n and its keywords $\{w_{n,1}, w_{n,2},...\}$ in a server, he encrypts the document and keywords with his group session key. For a member of a small group g_i in the jth session, the encrypted document and indexes in PKI-II are generated as follows;

$$\{f_{gk_i^j}(d_n), f_{gk_i^j}(D_n), f_{gk_i^j}(w_{n,1}), f_{gk_i^j}(w_{n,2}), \ \ldots\}$$

The user sends these to GM.

4.1.4 Database update

Receiving the encrypted document and its indexes, GM decrypts them with the group g_i's session key and then re-encrypts with the group search keys (index encryption key and document encryption key) and GM's secret key. Then, GM sends them to a server as follows:

$$\{f_{k_c}(d_n), f_{dk_j}(D_n), f_{ik_i}(w_{n,1}), f_{ik_j}(w_{n,2}), \ \ldots\}$$

The next process is the same as PKIS-I.

Table 2 Encrypted document

Identifier of documents	Encrypted document
$f_{k_c}(d_1)$	$f_{k_c}(f_{dk_1^1}(D_1))$
$f_{k_c}(d_2)$	$f_{k_c}(f_{dk_1^2}(D_2))$
...	...
$f_{k_c}(d_7)$	$f_{k_c}(f_{dk_3^1}(D_7))$
$f_{k_c}(d_8)$	$f_{k_c}(f_{dk_2^3}(D_8))$
$f_{k_c}(d_9)$	$f_{k_c}(f_{dk_3^2}(D_9))$
...	...
$f_{k_c}(d_{114})$	$f_{k_c}(f_{dk_{11}^{13}}(D_{114}))$
...	...
$f_{k_c}(d_{561})$	$f_{k_c}(f_{dk_8^{22}}(D_{561}))$
...	...
$f_{k_c}(d_n)$	$f_{k_c}(f_{dk_i^s}(D_n))$

4.2 Downloading phase

4.2.1 TrapGen(w, ik) construction

Main difference from PKIS-I in the construction of algorithm $TrapGen(w, ik)$ is that PKIS-II does not need to generate trapdoors as many as the number of session-changes. If a user wants to search a keyword w, the user encrypts the keyword with his group session key and sends the trapdoor to GM. Like the Database Update Stage, GM decrypts and re-encrypts them. Then, GM queries DS with it. For a member of a small group g_i, the trapdoor for a keyword w in PKIS-II is only one for every time like this;

$$T_w = (f_{ik_i}(w))$$

4.2.2 Retrieval(T_w) and Dec(E(D), dk) construction

The retrieval stage is also the same as PKIS-I. Receiving the results (encrypted documents) from DS, GM decrypts them with data encryption key dk_i and re-encrypts with group session key gk_i^j. And then, GM sends them to the user again. The user decrypts them with his group session key gk_i^j.

Figure 2 shows the whole process of PKIS-II.

5 Security Analysis

5.1 Group search secrecy

Our retrieval system is the group key-based cryptographic searching method on encrypted documents. Therefore, in this section, we discuss group key secrecy. The following are group key security requirements in [26].

- ○ **Group key secrecy**: It must be computationally infeasible for a passive adversary to discover any secret group key.
- ○ **Forward secrecy**: Any passive adversary being in possession of a subset of old group keys must not be able to discover any subsequent group key.
- ○ **Backward secrecy**: Any passive adversary being in possession of a subset of subsequent group keys must not be able to discover any preceding group key.
- ○ **Key independence**: Any passive adversary being in possession of any subset of group keys must not be able to discover any other group key.
- ○ Forward secrecy provides security for subtractive events (leave), since it prevents former group members from computing the updated group key. Similarly, backward secrecy provides security for additive events (join), because it prevents new members from discovering the previously used group keys [27].

In this paper, the term 'negligible function' refers to a function $\eta : \mathrm{N} \to \mathrm{R}$ such that for any $c \in \mathrm{N}$, there exists $n_c \in \mathrm{N}$, such that $\eta(n) < \frac{1}{n_c}$ for all $n \geq n_c$ [13].

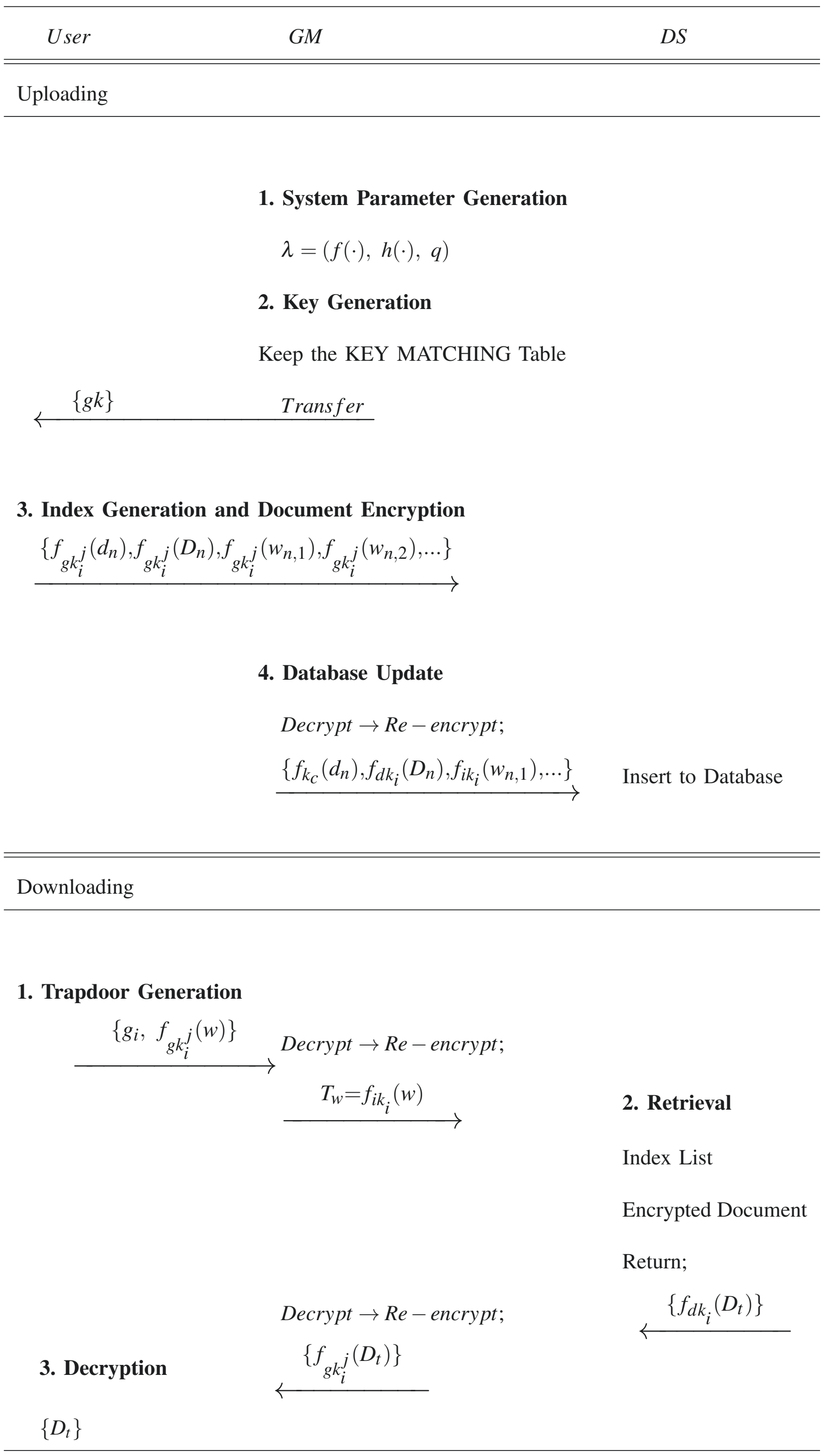

$User$
GM
DS
Uploading
1. System Parameter Generation
$\lambda = (f(\cdot),\ h(\cdot),\ q)$
2. Key Generation
Keep the KEY MATCHING Table
$\{gk\}$
$Transfer$
3. Index Generation and Document Encryption
$\{f_{gk_i^j}(d_n), f_{gk_i^j}(D_n), f_{gk_i^j}(w_{n,1}), f_{gk_i^j}(w_{n,2}), \ldots\}$
4. Database Update
$Decrypt \rightarrow Re-encrypt;$
$\{f_{k_c}(d_n), f_{dk_i}(D_n), f_{ik_i}(w_{n,1}), \ldots\}$
Insert to Database
Downloading
1. Trapdoor Generation
$\{g_i,\ f_{gk_i^j}(w)\}$
$Decrypt \rightarrow Re-encrypt;$
$T_w = f_{ik_i}(w)$
2. Retrieval
Index List
Encrypted Document
Return;
$\{f_{dk_i}(D_t)\}$
$Decrypt \rightarrow Re-encrypt;$
$\{f_{gk_i^j}(D_t)\}$
3. Decryption
$\{D_t\}$

Figure 2 The whole process of PKIS-II.

However, group key-based search system should not follow the above properties because a new joiner to the group such as a company or a government office should be able to search all of the previous documents to perform their successive tasks of the group. Namely, backward secrecy must not be a security requirement for our group search system. In this paper, we define group search secrecy as follows.

- **Forward search secrecy** : For any group g_i^j, the probability that a participant $p \in g_i^j$ can generate valid trapdoors for $(j+1)$th session is negligible when the participant knows valid group search key K_i^j, where $p \notin g_i^{j+1}$ and $0 < j < q$. ik_i^j and dk_i^j fall under K_i^j in PKIS-I and gk_i^j falls under K_i^j in PKIS-II.

It means that all leaving members from a group should not access to all of the next documents of the group any more.

- **Backward search accessibility** : For any group g_i^j, the probability that a participant $p \in g_i^j$ can generate valid trapdoors for $(j - l)$th session is $1 - \eta(n)$ when the participant knows valid group search key K_i^j, where $p \notin g_i^{j-l}$ and $0 < l < j$. ik_i^j and dk_i^j fall under K_i^j in PKIS-I and gk_i^j falls under K_i^j in PKIS-II.

Namely, all joining members to a group can access to all of the previous documents of the group.

- **Group search secrecy**: For a datacenter server DS, when a revelation of group search key K_i^j happens, the probability that DS can guess correctly the encrypted documents of group g_i at the jth session is negligible.

It must be computationally infeasible for DS to know or guess correctly the contents of the encrypted documents and trapdoors even if a leaving member or another member in a group reveals his group search keys.

5.1.1 PKIS-I

In PKIS-I, group search keys are reversely generated by the one-way hash key chain. Our scheme PKIS-I satisfies with Group Search Secrecy as follows.

- **Forward search secrecy**: By the Property 2 of Definition 1, if the latest released group search key is K_i^j, any participant cannot know a later value K_i^l such that $h^{l-j}(K_i^l) = K_i^j$. Therefore, the probability that a participant $p \in g_i^j$ can generate valid trapdoors for the next $(j+1)$th session is negligible, where $p \notin g_i^{j+1}$.
- **Backward search accessibility**: By the Property 1 of Definition 1, if the latest released group search key is K_i^j, any participant can deduce an earlier value K_i^l by applying the later value K_i^j to one-way hash key chain like this; $h^{j-l}(K_i^j) = K_i^l$. Therefore, the probability that a participant $p \in g_i^j$ can generate valid trapdoors for $(j - l)$th session is $1 - \eta(n)$, where $p \notin g_i^{j-l}$ and $0 < l < j$.
- **Group search secrecy**: In PKIS-I, GM re-encrypts all documents and indexes including trapdoors with his secret key k_c. Although one of group members reveals his/her group search keys to a datacenter server DS, DS cannot learn anything because DS does not know GM's secret key k_c. Therefore, the probability that DS can guess correctly the encrypted documents of group g_i at the jth session is negligible when K_i^j is revealed to DS.

5.1.2 PKIS-II

Group search keys ik and dk are unchangeable in PKIS-II and actual group search secrecy depends on group session key gk. When a user queries GM with a keyword, the keyword is encrypted by his/her group session key. If the user is a valid member of a certain group, GM can decrypt the querying keyword and then can generate a valid trapdoor for the user with his/her group search key. In this respect, it is proper that we regard a group session key as a group search key in PKIS-II. Thus, group search secrecy is up to the security of a group key agreement protocol.

- **Forward search secrecy**: If membership changes occur, a new group session key is generated and distributed securely to valid members according to a given protocol, and leaving members cannot get a new group session key. Hence, the leaving member cannot generate the valid trapdoor for a new session because GM decrypts a trapdoor with the group's newly updated session key.

We assume that a given group key agreement protocol satisfies with forward secrecy with the probability of $1 - \eta(n)$. Then, the probability that a participant $p \in g_i^j$ can generate valid trapdoors in the next $(j+1)$ session is negligible (or follows negligible function) when the participant knows the jth valid group search key $K_i^j (= gk_i^j)$.

- **Backward search accessibility**: For joining members, a new group session key is generated and distributed securely to valid members according to a given protocol, and the new joiners can also retrieve all of the previous documents because group search keys *ik* and *dk* are unchangeable in PKIS-II. If a joiner is authenticated as a valid user with his/her group session key, GM queries DS with a trapdoor instead of the user. The trapdoor is encrypted by unchangeable index generation key *ik*.

We assume again that the given group key agreement protocol satisfies with backward secrecy with the probability of 1 - η (n). Then, the probability that a participant $p \in g_i^j$ can generate valid trapdoors for (j -l) - th session is 1 - $\eta(n)$ when the participant knows valid group search key $K_i^j (= gk_i^j)$, where $p \notin g_i^{j-l}$ and $0 < l < j$.

- **Group search secrecy**: Members of a group cannot know their group search keys *ik* and *dk* in PKIS-II and only GM knows them. Even if a leaving member or another malicious member reveals his group session key *gk* to DS, DS cannot know the contents of the documents or trapdoor because they are encrypted with the group search keys *ik* and *dk* that group members do not know. Therefore, the probability that a datacenter server DS can guess correctly the encrypted data of a group g_i at the *j*th session is negligible when $K_i^j (= gk_i^j)$ is revealed to DS.

5.2 Keyword index search privacy

Song et al. [5] firstly proposed a cryptographic scheme which queries with encrypted keyword over encrypted data without decrypting anything by a server. They introduced four security requirements under an untrustworthy server. They are 'provable secrecy' (an untrustworthy server cannot learn anything about the plaintext given only the ciphertext), 'controlled searching' (an untrustworthy server cannot search for a word without the user's authorization), 'hidden queries' (an user may ask the untrustworthy server to search for a secret word without revealing the word to the server), and 'query isolation' (an untrustworthy server learns nothing more than the search result about the plaintext). However, Song's scheme is not for an index search system so that 'indistinguishability of indexes' have been considered additionally in other keyword index search schemes as well as the Song's requirements.

In our scheme, we assume an untrustworthy server as an adversary and our goal is to prevent a server from revealing or misusing users' information without users' consent. We accomplish our goal by encrypting documents and querying keywords. With relation to this goal, we define our security requirements using the term of 'Privacy'. The privacy is the ability to control private information, which includes identity and identifiers, and sensitive information [28], i.e., self-control for his/her information. The following is our definition about keyword index search privacy.

5.2.1 Retrieval access control

- User access control.

For participants $p \in g_i$, the probability that p can search for the documents of gt is negligible, where $i, t \geq 1$, $t \neq i$. It means that all of the users encrypt their documents with their secret key and can retrieve only their documents. It is because only a legitimate user who has a valid key can generate valid trapdoors and decrypt the retrieved data, where valid trapdoors mean the querying keywords to GM, generated by valid users.

1) PKIS-I: If a user $p \in g_i$ tries to retrieve some documents of a group g_t in the second session, p should know ik_t^1, ik_t^2 and dk_t^1, dk_t^2, which are encrypted with each group session keys and transferred to the group members of g_t like this: $f_{gk_t^2}(ik_t^2, dk_t^2)$, $f_{gk_t^2}(ik_t^2, dk_t^2)$. Refer to Figure 2. The only users that know the search keys ik_t^1 and ik_t^2 can generate valid trapdoors. Then, the users query GM with the trapdoors. Except for the members of a group g_t, nobody knows the values ik_t^2, ik_t^2 and dk_t^1, dk_t^2 because of the security of PRF f.

We assume that f is (t, q, e)-secure PRF and a user $p \in g_i$ tries to retrieve the documents of a group g_t in the jth session, where i, $t \geq 1$, $t \neq i$. Then, by Definition 2, we know $AdvA < e^j$, $0 < e < 1$. Therefore, we can say that the probability of retrieval is negligible.

In addition, if malicious leaving members from g_t reveal their group search keys to other groups' members when a session is changed from the second to the third, other users can know only ik_t^1, ik_t^2 and dk_t^1, dk_t^2. Because they cannot know new session's keys ik_t^3, dk_t^3, they cannot generate valid trapdoors for the third session so that they cannot be authenticated as valid users to GM.

This problem falls under Forward Search Secrecy.

2) PKIS-II: A user $p \in g_i$ should know gk_t^j to retrieve the documents of a group g_t in the jth session. This is because valid users generate trapdoors with their group session key and then query GM with the trapdoors in PKIS-II. The group session keys are distributed to the group members securely according to a given group

key agreement protocol. We assume that a given group key agreement protocol is secure for key distribution with the probability of 1 - $\eta(n)$. Therefore, the probability that a participant $p \in g_i$ can retrieve the documents of g_t follows negligible function $\eta(n)$, where $i, t \geq 1, t \neq i$.

- Server search control.

For a datacenter server DS, when DS generates trapdoors with a random selected keyword and search keys, the probability that a server succeeds in retrieving is negligible.

It is the similar concept to 'controlled searching' of [5] and 'capability' of [13]. An untrustworthy server cannot search for a word without given 'searching ability' from users. In our schemes, the concept is the same meaning as a valid trapdoor. The valid trapdoor generation requires that a user should know secret key values. Here, valid trapdoors mean the querying keywords generated by GM to a datacenter server DS.

1) PKIS-I: Valid trapdoors are generated by the secret values of each session in PKIS-I: an index generation key *ik* and GM's secret key k_c. The two values are secret keys for PRF *f*. By Definition 2, if DS generates trapdoors with a random selected keyword and search keys, the probability that a server can succeed in retrieving is e^2, negligible.
2) PKIS-II: Valid trapdoors are generated by an unchanging index generation key *ik*. In PKIS-II, *ik* is the secret key which any user does not know but only GM knows that. The key is also a secret key for PRF *f*. Therefore, by Definition 2, if DS generates trapdoors with a random selected keyword and search keys, the probability that a server can succeed in retrieving is *e*, negligible.

5.2.2 Unobservability

Generally, unobservability means that when a user utilizes a resource or service, the others cannot know the resource or service is being used [29]. If *f* is a pseudorandom function, *h* is one-way hash function, and all processes are performed according to the given protocol, all attackers(including insiders such as a datacenter server DS) cannot learn anything about the contents of encrypted documents by querying with encrypted keywords. It is because all the search processes by DS are implemented without decrypting anything.

We assume that *f* is (t, q, e)-secure PRF as we define earlier, *h* is (t, e_h) one-way hash function such that any attack algorithm *A* running in time *t* has success probability at most e_h, and a given group key agreement protocol is secure with the probability of 1 - $\eta(n)$. We choose the key material as described above, and all processes are done according to the given protocol. Then, our scheme PKIS-I can guarantee the security at least 1 - $\{e_h + (2e^2 + e) + e^2\}$ through whole processes in that an adversary cannot learn anything about the contents of encrypted documents except for the results.[e] PKIS-II can guarantee the security at least 1 - $\{\eta(n)+3e+2e\}$.

5.2.3 Unlinkability–index indistinguishability

Unlinkability means that when resources and services are used by someone, the others cannot link these being correlated or used together. In keyword index search system, it can be regarded as index indistinguishability.

Since Goh [8] formulated IND-CKA for indexes known as semantic security, most researchers have followed Goh's security definition and proof in this area. 'Indistinguishability for Indexes' guarantees that an adversary cannot deduce data's contents from its index list. An adversary cannot know even the fact whether two documents have the common keyword or not. Given two word lists W_0 and W_1, we say that the search scheme provides 'Index Indistinguishability' if a server S cannot distinguish the index list I_0 from I_1 for W_0 and W_1 with non-negligible advantage.

However, our schemes do not guarantee this property. In our scheme, the common keywords in different documents for a certain group have the same index values. Even if an adversary does not know what the keywords mean, the adversary can know that the keywords have something in common. An adversary might guess that two documents have something correlated. This is because we use only deterministic symmetric functions that have the same encryption value under the same data and the same key. And we did not use any random factor in our schemes. It makes our schemes more efficient than any other schemes because we can apply the database schema of 'primary key' and 'foreign key'. The details are addressed in the next section.

Consequently, our schemes can guarantee 'Retrieval Access Control' and 'Unobservability' but not 'Unlinkability'. However, in a common real world, users would like to choose practical schemes under the appropriate control of security other than the scheme which is hard to apply a real world due to inefficiency from the high level of security.

6 Experiments Of Performance

In this section, we describe the experiments of our proposed schemes.

6.1 Setting of experiments

Our system processes the transactions on an Intel Pentium 4 CPU 2.66 GHz processor with 512 MB RAM. We use MS SQL Server 2000 as the database system and use WinAPI C Library and MS-SQL DB Library for

C. These experiments use OpenSSL cryptography modules for cryptographic operations such as SHA-1 and AES. Table 3 describes the detailed implementation parameters. We assume different documents contain common keywords, and we set that a common keyword repeats at least every 435 documents among 10,000 documents.

Through our experiments, group search and efficiency can be identified as primary results of our schemes. Consequently, our experiments consist of largely two parts: Sections 6.2 and 6.3. Section 6.2 deals with the analysis of our schemes in group search. Section 6.3 deals with comparisons of our scheme PKIS-II with other schemes in order to show the efficiency of our schemes.

6.2 Analysis on PKIS-I and PKIS-II

We experiment with respect to the number of documents and the number of sessions. For example, the search process of PKIS-I takes about 7.9 ms (0.0079 s) at the first session and PKIS-II takes about 8.8 ms (0.0088 s) for 10,000 documents. Refer to Table 4. The main difference between PKIS-I and PKIS-II is key management.

In PKIS-I, group search keys *ik* and *dk* are reversely generated with hash key chains by GM, which are dynamic to session-changes. The group search keys for each session are encrypted with a group session key and then transferred to group members. Actual encryption keys for indexes and documents in database tables are made up of the group search keys and GM's secret key. This means that secret values are managed together by group members and GM. Especially, the more number of sessions have passed, the more trapdoors for one keyword query should be generated in PKIS-I, because group search keys *ik* and *dk* are updated dynamically to session-changes. Nevertheless, the searching time of PKIS-I is only within 53 ms (0.053 s) when a session is the 1000th. In fact, the current session may be over 1000 in some environments such as mobile environments, and it would require more time and computational overheads. However, our applications are for organizations such as companies or municipal offices, so that our performance can manage these applications (group organizations) sufficient.

In PKIS-II, group search keys *ik* and *dk* are unchanging irrespectively of session-changes. GM keeps a key matching information for groups, where group search keys *ik* and *dk* are matched to the dynamic group's session keys. When group members query GM with some data, the data should be encrypted with the group's session key, whereby a group member can be authenticated as a valid group member. Once a member passes the authentication, most processes are implemented by GM instead of the member. Receiving some data from a group member or a server, GM decrypts and re-encrypts the received data, so that GM gets to know all of the contents of documents and trapdoors every query time. However, only one trapdoor is sufficient for one keyword due to unchanging group search keys independently of session-changes. The invariable searching time is required irrespectively of session-changes. If the current number of session is high, the performance of PKIS-II is more efficient than PKIS-I as described in Table 4.

Table 4 Searching time according to session-changes (time unit: ms)

Scheme	PKIS-I				PKIS-II
No. of sessions	1	10	100	1000	
2500 documents	5.8	8.0	9.9	38.6	6.8
5000 documents	6.9	10.3	13.9	42.6	7.8
7500 documents	7.4	13.9	16.3	49.3	8.4
10000 documents	7.9	13.9	18.3	52.7	8.8

Table 3 Implementation environment and parameters

Agent	Processor	Intel Pentium 4 CPU 2.66 GHz
	RAM	512 MB
	Language	C++
	Crypto. Eng.	OpenSSL Crypto Library(AES-CBC-128)
Database	Product	MS SQL Server 2000
	Interface	WinAPI
	Library	MS-SQL DB Library for C
Cryptographic	PRF	AES (128 bits)
Parameter	Hash function	SHA-1 (160 bits)
	The number of keywords	7
Dataset	The number of common keywords	≥ 435
	The number of documents	2500 = 5000 = 7500 = 10000
	The number of sessions	1 = 10 = 100 = 1000

6.3 Comparison of our scheme with other schemes

6.3.1 The results of implementation

In order to evaluate our scheme's performance with objective validity, we experiment the following four previous schemes: (1) Song et al.'s [5]; (2) Golle et al.'s [13]; (3) Waters et al.'s [9] variation; and (4) Park et al.'s [30]. Song et al. deal with the symmetric cryptographic method as a pioneering work in this area. Golle et al. conduct the most secure scheme, which satisfies 'query isolation' and index indistinguishability as well. Waters et al. deal with audit log server; however, we assume that their server is as a general database server, because their keyword search technique on the encrypted data has wide applications beyond searchable audit logs. We experiment only one, symmetric scheme of their two symmetric and asymmetric schemes, because symmetric scheme is much faster. Park et al.'s schemes also deal with symmetric methods. They work on similarity search, and their schemes are the encrypted characters by characters. The searching method is approximate string matching test by hamming distance, i.e., we can expect the schemes would be inefficient. However, Park et al. maintain Golle et al.'s security and improve Golle et al.'s inefficiency in spite of the characterwise encryption method. In their paper [30], they did not show the formal security proof and the experimental proof. Therefore, this paper compares Golle et al.'s and Park et al.'s with our schemes.

Although there are many papers as the recent schemes such as [18,20-23], [18,20,21] do not deal with the Boolean operation on keyword searches as the traditional searchable encryption schemes, but the ranked search operation. As we mentioned earlier, the comparison with our method is meaningless, because their evaluation method and security requirements are different. In addition, these schemes of [22,23] are also not appropriate to compare with our schemes, because [22,23] deal with asymmetric schemes based on pairing-based cryptography. Section 6.3.3 demonstrates the detailed reasons.

In order to evaluate the efficiency of encrypted search systems more precisely, we also perform experiments on the plaintext version (PKISIIP) without encryption. We compared only PKIS-II with other schemes, because our schemes take the multiple user setting of group search. On the other hand, PKIS-II has the similar search processes to other schemes, because it does not require the group search key changes such as PKIS-I.

Table 5 shows the result of our experiments. The performance of our scheme is much better than the existing schemes. For instance, the performance of PKIS-II is about 935 times faster than Golle's scheme and about 16 times faster than Song et al.'s scheme for 10,000 documents. Park et al.'s schemes, SSS-I and SSS-II are not fast but their schemes are faster than Golle's as they claimed.

Table 5 Searching time comparison with other schemes (time unit: ms)

	Song	Golle	Waters	SSS-I	SSS-II	PKIS-II	PKIS-IIP
2500 documents	47	2094	79	270	1721	6.8	2.8
5000documents	62	4439	157	536	3269	7.8	3.8
7500documents	109	6991	204	814	5088	8.4	4.3
10000documents	147	8229	297	969	6756	8.8	4.8

In the search process, PKIS-II needs very slight computational overheads, within 10 ms (0.01 s). With the respect to time consumption, a search process is the most important factor. The search process of PKIS-II is similar to general plaintext search system because it can directly access the data without verifying for every row. It needs the additional time only to generate a trapdoor and to decrypt returned documents. The used cryptographic function in PKIS is also very fast.

From the next subsection, we analyze our results in two respects of the applicability of DB schema and the influence of functions.

6.3.2 The applicability of DB schema

In most existing schemes, the indexes of each document are encrypted with random factors for indistinguishability and the encrypted indexes are stored by a row. Hence, a server should implement at least one computation for each document every row to verify whether this document contains the querying keyword or not. This makes it difficult to apply DB schemas into encrypted database search systems. Accordingly, the computational complexity of previous schemes requires at least $O(n)$ if the number of documents is n. In addition, most previous schemes store a document's indexes by a row not in a field (column). The computation or scanning within one field is relatively faster than within one row. In contrast, the computation or scanning for many fields within one row is not fast.

Our schemes solved these problems by different database structures from other schemes. In Table 1 Index List, all of the indexes for all documents are stored in one field. Generally, the row size limitation is strict but the field size of database is at least 4 TB or more, i.e. relatively unrestricted. For example, the maximum number of bytes per row of MS SQL 2000 is only 8 kB and MS SQL 2005 is 2 GB [31]. Hence, setting an index column for all indexes does not have any problem in our schemes, and the encrypted documents and their identifiers are stored in another table.

We achieved database ‘normalization’ with ‘primary key’ and ‘foreign key’. This is possible because we use different database table structure and deterministic functions. We do not use any random factors. Consequently, these properties enable a server to directly access the data that a user wants. Thus, there is no computation to test whether this document contains the querying keyword or not for every row.

6.3.3 The influence of function

The kind of applied functions greatly influences on the search time. There are many schemes dealing with bilinear function such as [13,22,23,32-37] among the recently proposed keyword search schemes. For example, in the experiment of [35], searching 10,000 indexes requires approximately 720 s (720000 ms). Compared with symmetric cryptographic method, the calculation of one pairing takes much more time. Consequently, bilinear function is not appropriate for real-world applications. On the other hand, our proposed schemes are based on the only symmetric cryptographic function.

7 Conclusion

In cloud computing environments, DAS model is the most realistic to manage sensitive information with safety, because a server manager is considered untrustworthy. Encryption over database is also one of the most substantial ways in order to accomplish the goal of the DAS model. Although the encryption method has some negative effects such as inefficiency and hardness of applying DB schemas, we should not hinder the performance or general operations of database because of the encryption for security and privacy.

Considering prior researchers’ endeavors in the individual setting between a server and a user, this paper focuses on more realistic applications and environments with two aspects: the group search and efficiency. To do this, firstly, we conduct a group search rather than a private setting. This group search does not require re-encrypting all documents under the key update from session-change. Secondly, for more efficient application in a real world, we develop the database table in order to apply the efficient DB schemas (normalization using primary key and foreign key) to encrypted documents. Also, we define and analyze the group search secrecy and keyword index search privacy. Moreover, this paper represents our scheme’s efficiency through experiments.

This paper realizes efficient performances by developing two novel encrypted database tables. These two encrypted database tables make it possible a server to access data directly. Prior papers’computational complexity is at least $O(n)$, while our schemes’ computational complexity is $O(1)$ during a search process. Therefore, our scheme is approximately 935 times faster than Golle’s scheme and around 16 times faster than Song’s scheme for 10,000 documents.

As the result of our experiments, we maintain the characteristics of DB application layers, which supports the interoperability of DB applications in order to design efficient schemes. This paper has two contributions: (1) in the cloud datacenter service environments, our schemes provide practical and realistic encrypted DB solution and (2) identifying the importance of interoperability with DBMS for designing efficient schemes.

For future works, we need to focus on the more experiments of the performance in real mobile applications. In cloud computing environments, end-users require various types of usages with mobile applications such as PDA or mobile phone as many as PCs. Therefore, we believe ‘interoperability’ of a mobile application and ‘compatibility’ between mobile and DB applications as important factors to improve the efficiency of schemes.

9 Endnotes

[a]DB schema is the structure of a database system, described in a formal language supported by the DBMS. In a relational database, the schema defines the tables, the fields in each table, and the relationships. [b]Database normalization can be defined as the practice to optimize table structures. Particularly concentrating on how these data are interrelated, optimization is the result of a investigation from the various pieces of data stored within the database. Considering the analysis of this data and its corresponding relationships, it is advantageous in two points: first, the analysis will be the result of substantial improvement of the speed when the tables are queried; second, it decreases the chance of the database integrity compromised due to tedious maintenance procedures. [c]In ranked search, term frequency means a count of the number of times that term appears in that document [16]. [d]The perspective of utility computing. The cloud computing technologies and services enables for providers and companies to offer a policy: pay-for-what-you-use such as that of electricity, fuel, and water. With these economic strengths, cloud computing has become a leading computing technology and expanded seamless services; however, security studies encounter new challenges and issues in cloud computing era. First of all, the datacenter of cloud storage services has high risk of information leakage by intruders or insiders. Especially, it cannot guaranteed that datacenter managers are trustful. Storing confidential information outside (datacenter) makes the data center risky in terms of the infringement of privacy and security. Cloud services are broadly divided into three categories: Infrastructure-as-a- Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) [38]. [e]The first part within

a brace is for key generation, the second part is for database table, and the third part is for trapdoor.

Acknowledgements
This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-C1090-1001-0004).

Author details
[1]Graduate School of Information and Security, Korea University, 5-Ka, Anam-dong, Sungbuk-ku, Seoul 136-701, Korea [2]Department of Information Systems, Weatherhead School of Management, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106, USA

Competing interests
1) The article-processing charge for this manuscript was supported by "ITRC" support program supervised by the NIPA (National IT Industry Promotion Agency) of Korea.
2) The two encrypted tables to build database normalization is 'Korean Registered Patent' by Hyun-A Park, Registered No.(10-0839220), June. 11. 2008.

Received: 15 December 2010 Accepted: 17 August 2011
Published: 17 August 2011

References
1. M Armbrust, A Fox, R Griffith, AD Joseph, RH Katz, A Konwinski, G Lee, Above the clouds: a Berkeley view of cloud computing. *Technical Report: EECS-2009-28* (February 10, 2009)
2. R Buyya, Market-oriented cloud computing: vision, hype, and reality of delivering computing as the 5th utility, in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, ccgrid*, 1 (2009)
3. H Park, J Byun, D Lee, Secure index search for groups, in *TrustBus 2005, LNCS3592*, 128–140 (2005)
4. P Wang, H Wang, J Pieprzyk, Threshold privacy preserving key word searches, in *SOFSEM 2008, LNCS 4910*, 646–658 (2008)
5. D Song, D Wagner, A Perrig, Practical techniques for searches on encrypted data, in *IEEE Symposium on Security and Privacy*, 44–55 (2000)
6. D Boneh, GD Crescenzo, R Ostrovsky, G Persiano, Public-key encryption with keyword search, in *Eurocrypt04, LNCS 3027*, 506–522 (2004)
7. YC Chang, M Mitzenmacher, Privacy preserving keyword searches on remote encrypted data. Cryptology (ePrint Archive) (2004)
8. E Goh, Secure indexes. Cryptology (ePrint Archive) (2004)
9. B Waters, D Balfanz, G Durfee, D Smetters, Building an encrypted and searchable audit log, in *NDSS04, The Internet Society*, 205–214 (2004)
10. J Byun, H Rhee, H Park, D Lee, "Off-Line Keyword Guessing Attacks on Recent KeywordSearch Schemes over Encrypted Data", in *SDM2006, Lecture Notes in Computer Science 4165*, 75–83 (2006)
11. D Boneh, B Waters, Conjunctive, subset, and range queries on encrypted data, in *Proceedings of TCC 07* (2007)
12. H Hacigumus, B Iyer, S Mehrotra, Efficient execution of aggregation queries over encrypted relational databases, in *DASFAA 2004, LNCS 2793*, 125–136 (2004)
13. P Golle, J Staddon, B Waters, Secure conjunctive keyword search over encrypted data, in *ACNS04, LNCS 3089*, 31–45 (2004)
14. Y Hwang, P Lee, Public key encryption with conjunctive keyword search and its extension to a multi-user system, in *Pairing 2007, LNCS 4575*, 2–22 (2007)
15. P Wang, H Wang, J Pieprzyk, Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups, in *CANS 2008, LNCS* (2008)
16. S Zerr, E Demidova, D Olmedilla, W Nejdl, M Winslett, S Mitra, Zerber: r-confidential indexing for distributed documents, in *EDBT'08: Proceedings of the 11th international conference on Extending database technology*, 287–298 (2008)
17. S Zerr, D Olmedilla, W Nejdl, W Siberski, Zerber+R: top-k retrieval from a confidential index, in *EDBT '09: Proc. of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, 439–449 (2009)
18. H Pang, X Ding, X Xiao, Embellishing text search queries to protect user privacy, *PVLDB 3(1)*, 598–607 (2010)
19. A Swaminathan, Y Mao, G-M Su, H Gou, A Varna, S He, M Wu, D Oard, Confidentiality-preserving rank-ordered search, in *Storage SS'07, in Proc. of the 2007 ACM workshop on Storage security and survivability*, 7–12 (2007)
20. C Wang, N Cao, J Li, K Ren, W Lou, Secure ranked keyword search over encrypted cloud data, in *ICDCS'10, in Proc. of the 2010 IEEE 30th International Conference on Distributed Computing Systems*, 253–262 (2010)
21. N Cao, C Wang, M Li, K Ren, W Lou, Privacy-preserving multikeyword ranked search over encrypted cloud data, in *IEEE INFOCOM* (2011)
22. M Li, S Yu, N Cao, W Lou, Authorized private keyword search over encrypted data in cloud computing, in *Proc of IEEE ICDCS'11* (2011)
23. S Yu, C Wang, K Ren, W Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing, in *IEEE INFOCOM'10* (2010)
24. Y Hu, A Perrig, DB Johnson, Efficient security mechanisms for routing protocols, in *Network and Distributed System Security Symposium, NDSS'03*, 57–73 (February 2003)
25. M Burrnester, Y Desmedt, A secure and efficient conference key distribution system, *The Advances in Cryptology–EUROCRYPT* (1994)
26. Y Kim, A Perrig, G Tsudik, Tree-based group key agreement. ACM Trans Inf Syst Secur. **7**(1), 60–96 (2004). doi:10.1145/984334.984337
27. L Liao, M Manulis, Tree-based group key agreement framework for mobile ad-hoc networks. Fut Gener Comput Syst. **23**(6), 787–803 (2007). doi:10.1016/j.future.2007.01.001
28. M Burmester, Y Desmedt, RN Wright, A Yasinsac, Accountable Privacy. *Security Protocols 2004*, LNCS 3957, 83–95 (2006)
29. Ontario, Office of the Information and Privacy Commissioner (IPC) and Netherlands Registratiekamer. Privacy-Enhancing Technologies: The Path to Anonymity, Information and Privacy Commissioner and Registratiekamer http://www.ipc.on.ca/english/Resources/Discussion-Papers/Discussion-Papers-Summary/Default.aspx?id=329&print=1 (1995)
30. H Park, B Kim, D Lee, Y Chung, J Zhan, Secure similarity search, *Grc 2007*, (IEEE ComputerSociety Press, 2007), pp. 598–604
31. http://blogs.msdn.com/msdnts/archive/2006/12/01/row-size-limitation-in-sql-2000-and-2005.aspx
32. M Abdalla, M Bellare, D Catalano, E Kiltz, T Kohno, T Lange, J Malone-Lee, G Neven, P Paillier, H Ashi, Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. *Crypto05, LNCS 3621* 205–222 (2005)
33. S Bellovin, W Cheswick, Privacy-enhanced searches using encrypted bloom filters. *Cryptology ePrint Archive, R eport2004/022* (February 2004)
34. L Ballard, M Green, B de Medeiros, F Monrose, Correlation-resistant storage via keyword-searchable encryption, in *SPAR Technical Report*. TR-SP-BGMM-050705
35. L Ballad, S Kamara, F Monrose, Achieving efficient conjunctive keyword searches over encrypted data. in *ICICS 2005, LNCS3783*, 414–426 (2005)
36. W Ogata, K Kurosawa, Oblivious keyword search. J Complexity. **20**, 356–371 (2004). doi:10.1016/j.jco.2003.08.023
37. H Park, J Hong, J Park, J Zhan, D Lee, Combined authentication based multi-level access control in mobile application for DailyLifeService. IEEE Trans Mobile Comput. **9**(6), 824–837 (2010)
38. H Park, J Park, J Choi, D Lee, Toward an integrated system between cloud computing and smartcard application, in *ICCIT 2010* (IEEE Computer Society Press, 2010), pp. 580–587

doi:10.1186/1687-1499-2011-64
Cite this article as: Park *et al.*: **PKIS: practical keyword index search on cloud datacenter.** *EURASIP Journal on Wireless Communications and Networking* 2011 **2011**:64.