A Multi-client Dynamic Searchable Symmetric Encryption System with Physical Deletion

Lei Xu[1,2], Chungen Xu[1(✉)], Joseph K. Liu[2], Cong Zuo[2], and Peng Zhang[3]

[1] School of Science, Nanjing University of Science and Technology, Nanjing, China
xuleinjust@yeah.net, xuchung@njust.edu.cn

[2] Faculty of Information Technology, Monash University, Melbourne, VIC, Australia
{joseph.liu,cong.zuo1}@monash.edu

[3] ATR Key Laboratory of National Defense Technology,
College of Information Engineering, Shenzhen University, Shenzhen, China
zhangp@szu.edu.cn

Abstract. Dynamic Searchable Symmetric Encryption (DSSE) provides a simple and fast storage as well as retrieval method for encrypted profiles which stored in cloud. However, due to the nature of the symmetric encryption algorithm, it allows only one client to access the data. To make the scheme more practical, this paper propose a multi client dynamic symmetric searchable encryption scheme that could allow multi-client to search the privacy data with the delegation search token and dynamic delete expected files with delete token. Compared with similar works, our construction achieves a balance in network security and practical performance. We also demonstrate that the proposed scheme has same IND-CKA2 security property against adaptive adversary.

Keywords: Searchable symmetric encryption · Cloud storage
Multi-client · RSA function

1 Introduction

Cloud storage is a new concept that extends and develops in the concept of cloud computing, which collects data storage and service access functions through the combination of cluster application, network technology or distributed file system, and collects many different types of storage devices in the network together through application software to work together. It is an emerging network storage technique, which has many good qualities. For one thing it makes all the storage resources be integrated together to achieve data storage management automation and intelligence, for another, it improve the storage efficiency and flexible expansion through the visualization technology to solve the waste of storage space, reduce the operating costs [1–4]. Due to its properties of flexible management and low rental prices, many users and businesses choose to put their own data in the cloud.

https://doi.org/10.1007/978-3-319-89500-0_45

With the promotion of cloud services, the industry soon found that when cloud storage brings people convenience, it gradually appears some short boards, and the biggest obstacle of cloud service promotion is the security issues around the data. The user suspects that the cloud service can not provide the corresponding security support for the data, which hinders the transfer of more data and business platform. In order to solve the problem mentioned above, we need to satisfies the following two conditions: Integrity and confidentiality, that is the cloud storage server should ensure that the data and operations in the cloud would not be malicious or non-malicious loss, destruction, leakage or illegal use; Access and privacy, when users visit some sensitive data, the system can prevent potential rivals to infer the user's behavior through the user's access mode. At present, the main means to solve such problems is to use the cryptography techniques. Users always use encryption system to encrypt their sensitive data before upload to the cloud to protect the data's confidentiality from illegal adversary. This method is the most straightforward and the simplest, but is not practical in the real scene. After a long period of research, for the former, people find that the searchable encryption is good tool to solve this problem. In this paper, we will focus on how to realize the dynamic data confidentiality and privacy retrieval control in cloud.

1.1 Related Work

The earliest research on searchable symmetric encryption system can be traced back to Chor et al.'s work [5] in 1995. They proposed the first retrieval scheme on encrypted data that stored in the database, which enables the user to search the special encrypted file without leaking anything of the data. After Chor's work, searchable symmetric encryption has been deeply studied and most of them focus on improving search performance, search pattern and security [6–8]. Cash et al. [9] renewed the encrypted data structure refer to the original one, and designed the first sub-linear SSE scheme which supported boolean queries for large databases at the cost of leaking the search pattern to the server. To make up for the lack of that Cash's work can only support single search, Jarecki extended Cash's OXT protocol to multi-client OXT [10] through provided the client s set of partial trapdoors for some permitted keywords. Their core policy is to define a sequence of attributes corresponding each query on an element in the keyword set, and the token could be computed when it satisfy the attributes.

There are also a lot of other works focus on realizing the dynamic search model, multi-client searching and other functions [11–15]. Compared with SE supporting single user, which can be regarded as data outsourcing, multi-user SE can achieve share of sensitive data. Generally, many existing SE schemes use key sharing, key distribution, proxy re-encryption, broadcast encryption, or other techniques to achieve the extension from single user to multi-user. Such as, in 2006, Curtmola et al. [16] proposed the first multiuser SE system under a broadcast encryption system, which brings enormous cost of user revocation. In 2008, Bao et al. [17] also proposed a multi-user SE. Because the users access rights depend on corresponding attribute set, the efficiency of system will increase by number of users. Dong et al. [18] constructed multi-user system based on proxy re-encryption techniques, where each user has its own unique key to encrypt, search and decrypt data. Thus, the scheme need a trusted server to manage keys. At the same time, recently, there are many systems based on ABE, in which user used attribute set to define rights of search [19–22]. Wang et al. [19] achieves fine-grained access control to authorized users with different access rights using a standard CP-ABE without key share. 2016, Wang et al. [20] proposed an efficiently multiuser searchable attribute-based encryption scheme with attribute revocation and grant for cloud storage. In the scheme, attribute revocation and grant processes of users are delegated to proxy server. In 2015, Rompay et al. [23] introduces a third party, named a proxy, that performs an algorithm to transform a single user query into one query per targeted document. In this way, sever cannot have access to content of query and its result, which achieves query privacy.

1.2 Our Contribution

In this work, we provide a multi-client dynamic searchable symmetric encryption system (MC-DSSE) for retrieving encrypted privacy data in cloud, and the main properties are listed as follows:

1. Multi-client. For practical use, this work focus on achieving single-writer/multi-reader search mode. It allows the data owner to delegate the search capability to multi-clients by a RSA approach. In fact, we distinguish the client by giving them the different ability search for a set of permitted keywords. When someone want to search for some special keyword, he needs to apply a partial search token from the data owner firstly, then generates the full search token according to the expected keyword.
2. Dynamic. To enhance the flexibility of the scheme, we add the **AddKeyword**, **DeleteFile**, algorithm to make it dynamic for the data owner. With these algorithms, the data owner can use his private key to add the new keyword and delete the encrypted file with the delete token.
3. Privacy. The proposed scheme achieves IND-CKA2 secure against probability polynomial adversary. Users could search the encrypted data stored in cloud platform which contains some keywords by a unique token without leaking anything about the origin data. Moreover, we also demonstrate that our scheme is secure for multi-clients by employ the RSA function.

1.3 Organization

The rest of this paper is organized as follows: In Sect. 2, we describe the definition of MC-DSSE scheme and gave some hardness assumptions. In Sects. 3 and 4, we propose a novel DSSE scheme support for multi-client and give its security proof. Section 5 gives its communication and computation cost. Finally, we end the paper with a brief conclusion.

2 Preliminaries

In this section, we first review the definition of the multi-client dynamic searchable symmetric encryption with keyword search, and then introduce some hardness problems with its complexity assumption related to our security proof.

2.1 MC-DSSE Definition and Related Database Structure

Here mainly introduce the syntax of multi-client dynamic symmetric searchable encryption and give a brief description of some necessary database structures.

Definition 1 (MC-DSSE) [24]**.** *A MC-DSSE scheme consists of the following five polynomial algorithms among a data owner, a client and a server:*

- ***Setup:****The data owner takes security parameter* λ *and a database* DB *as input, generate the system master key* MK *and public key* PK*, and sends the encrypted database* EDB *to the server, the server stores* EDB*.*
- ***ClientKGen:*** *The data owner takes* MK*, and a set* w *of permitted keywords as input and generates a search authorized private key sk for the client.*
- ***AddKeyword:*** *The data owner takes a new the file-keyword pair* (id, w) *and his secret key as input, generates and sends the ciphertexts to the server. The server takes the* EDB *as input, and inserts these ciphertexts into* EDB*.*
- ***DeleteFile:*** *The data owner takes the file's identifier and his secret key as input, returns a delete token to the server. The server takes the* EDB *as input, and deletes all ciphertexts of a file with identifier id from* EDB*.*
- ***Search:*** *The client takes the keyword and his secret parameters as inputs, generates a search token for the server. Then the server takes the database* EDB *as input, and returns the corresponding file identifiers of the file.*

In order to make the proposed scheme look more concise and practical, here it will employ two data structures $\mathcal{D}, \mathcal{T}$ which denotes List and Dictionary respectively, and then introduce four database language **Great, Get, Update**, **Remove** from [24], and it also will be used in our construction.

2.2 Security Definition and Hardness Assumptions

In this paper, we consider IND-CKA2 security of our MC-DSSE scheme. First, we define four response rules for the simulator for returning each query (Such as **Setup**, **AddKeyword**, **DeleteFile**, **Search**) of adversary $\mathcal{A}$, which will be used in our security model, and then give detail IND-CKA2 security model for our multi client searchable encryption.

- When $\mathcal{A}$ gives a selected database DB to $\mathcal{S}$ to have a test on protocol **Setup**, $\mathcal{S}$ takes leakage function $\mathcal{L}_{Setup}$ as input, and simulates an encrypted database EDB.
- When $\mathcal{A}$ gives a new file-keyword pair to $\mathcal{S}$ to have a test on protocol **AddKeyword**, $\mathcal{S}$ takes leakage function $\mathcal{L}_{AddKeyword}$ as input, and generates the corresponding searchable ciphertexts.

- When $\mathcal{A}$ gives a selected file to $\mathcal{S}$ to test on protocol **DeleteFile**, $\mathcal{S}$ takes leakage function $\mathcal{L}_{DeleteFile}$ as input, and generates the corresponding delete token.
- When $\mathcal{A}$ gives a selected keyword to $\mathcal{S}$ to have a test on protocol **Search**, $\mathcal{S}$ takes leakage function $\mathcal{L}_{Search}$ as input, and simulates the corresponding search token.

Definition 2 (IND-CKA2 Security) [24]**.** *Let Π = (**Setup, AddKeyword, DeleteFile, ClientKGen, Search**) be a multi client dynamic symmetric searchable encryption scheme, $\mathcal{A}$ and $\mathcal{S}$ denote the adversary and simulator, respectively. Suppose tuple ($\mathcal{L}_{Setup}$, $\mathcal{L}_{AddKeyword}$, $\mathcal{L}_{DeleteFile}$, $\mathcal{L}_{ClientKGen}$, $\mathcal{L}_{Search}$) be five leakage functions, consider the related two probabilistic games as follows:*

Real$_A(1^k)$*: $\mathcal{A}$ chooses an initial database* DB*. A challenger runs **Setup** to generate* (MK, PK, EDB) *where* (PK, MK) *denote the public/secret key of data owner and* DB *denotes the encrypted data of database* DB*. Once $\mathcal{A}$ receives the* EDB *from challenger, it makes a polynomial number of queries for protocol **AddKeyword**, **DeleteFile**, **ClientKGen** and **Search**. For response, the challenger feedbacks the corresponding result to $\mathcal{A}$. Finally, the adversary $\mathcal{A}$ outputs a bit 'b' as the result of the game.*

Ideal$_{A,S}(1^k)$*: $\mathcal{A}$ chooses an initial database* DB*. Given the leakage $\mathcal{L}_{Setup}$, $\mathcal{S}$ computes and sends encrypted database* EDB *to $\mathcal{A}$. Then $\mathcal{A}$ makes a polynomial number of queries for the five protocols as above. For each query, $\mathcal{S}$ masters the relevant leakage function five-tuple ($\mathcal{L}_{AddKeyword}$, $\mathcal{L}_{DeleteFile}$, $\mathcal{L}_{ClientKGen}$, $\mathcal{L}_{Search}$), For response, the challenger feedbacks the corresponding result to $\mathcal{A}$. Finally, the adversary $\mathcal{A}$ outputs a bit 'b' as the result of the game.*

We say that a multi-client DSSE scheme is called IND-CKA2 secure with leakage functions above, if the probability $Pr[\mathbf{Real}_A(k) = 1] - Pr[\mathbf{Ideal}_{A,S}(k) = 1]$ is negligible for some security parameter k.

Definition 3 (Strong RSA Problem) [25]**.** *Let p, q be two k-bit big prime numbers, and set $n = pq$. Choose $g \in \mathbb{Z}_n^*$ randomly. We say that an efficient algorithm $\mathcal{A}$ solves the strong RSA problem if it receives as input the tuple (n, g) and outputs two element (z, e) such that $z^e = g \mod n$.*

3 Our MC-DSSE Construction

Assume Data owner, Server, Client be the participants who take part in the DSSE scheme. With the four database language described in Sect. 2.1, now we design our detail multi-client dynamic searchable symmetric encryption scheme which includes the following five phases.

Setup$(1^k$, DB, $NULL)$**:**

- Data owner: Take a security parameter k and a database DB as inputs. Let F:$\{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^k$ be a key-based pseudo random function, and

H: $\{0,1\}^* \rightarrow \{0,1\}^{2k+1}$, G: $\{0,1\}^* \rightarrow \{0,1\}^{3k+1}$ be two cryptographic hash functions. Choose two big prime integers p, q, and pick $k_1, k_2 \in \{0,1\}^k$ randomly, then output the master key MK $= (p, q, k_1, k_2, g)$ and the public key PK$= (n = pq, F, G, H)$. Finally, run the Algorithm 1 to generate the EDB and send it to the Server, keep the $\mathcal{T}_P$ secret.
– Server: Store the encrypted database EDB.

Algorithm 1. EDB Generate Algorithm

```
Require: MK, PK, DB;
Ensure: $\mathcal{I}$
 1: EDB $\leftarrow \{\}$, $\mathcal{T}_P \leftarrow \{\}, \mathcal{T}_W \leftarrow \{\}, \mathcal{T}_F \leftarrow \{\}$, $\mathcal{T}_{F,W} \leftarrow \{\}$
 2: for $w \in$ DB do
 3:     if $P_w = NULL$ then
 4:         $L_w \leftarrow F_{k_1}(g^{1/w} \mod n)$
 5:     else
 6:         $L_w \leftarrow P_w$
 7:     end if
 8:     for $id \in$ DB$(w)$ do
 9:         if $P_{id} = NULL$ then
10:             $L_{id} = F_{k_1}(g^{1/id} \mod n)$
11:         else
12:             $L_{id} \leftarrow P_{id}$
13:         end if
14:         $\mathcal{T}_P \leftarrow \mathcal{T}_P \cup (id, P_{id})$, $L_{id,w} = F_{k_1}(g^{\frac{1}{id \cdot w}} \mod n)$
15:         $R_w \xleftarrow{\$} \{0,1\}^k$, $D_w \leftarrow H(F_{k_2}(g^{1/w} \mod n), R_w) \oplus (0||id||P_w)$
16:         $K_1 \leftarrow (L_w, D_w\ , D_w)$, $\mathcal{T}_W \leftarrow \mathcal{T}_W \cup K_1$
17:         $R_{id} \leftarrow \{0,1\}^k$, $D_{id} \leftarrow G(F_{k_2}(g^{1/id} \mod n), R_{id}) \oplus (0||L_w||L_{id,w}||P_{id})$
18:         $K_2 \leftarrow (L_{id}, D_{id}, D_{id,2})$, $\mathcal{T}_F \leftarrow \mathcal{T}_W \cup K_2$
19:         $R_{id,w} \leftarrow \{0,1\}^{2k}, D_{id,w} = H(F_{k_2}(g^{\frac{1}{id \cdot w}} \mod n), R_{id,w}) \oplus (0||L_w||L_{id})$
20:         $K_3 \leftarrow (L_{id,w}, D_{id,w,1}, D_{id,w,2})$, $\mathcal{T}_{W,F} \leftarrow \mathcal{T}_{W,F} \cup K_1$
21:     end for
22:     $\mathcal{T}_P \leftarrow \mathcal{T}_P \cup (w, P_w)$,
23: end for
24: return EDB $\leftarrow (\mathcal{T}_W, \mathcal{T}_F, \mathcal{T}_{F,W})$, $\mathcal{T}_P$
```

ClientKGen(MK, $\mathbf{w}$):

– Client: Assuming that a legitimate client wish to perform searches over keywords $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, he send $\mathbf{w}$ to the owner to apply for his private key of keywords $\mathbf{w}$.
– Data owner: The data owner generates a corresponding private key as:

$$sk_{\mathbf{w}} = (sk_{\mathbf{w},1}, sk_{\mathbf{w},2}, sk_{\mathbf{w},3}) \leftarrow (k_1, k_2, g^{1/\prod_{j=1}^{n} w_j} \mod n)$$

and then sends back $sk_{\mathbf{w}}$ together with $\mathbf{w}$ to the client.

AddKeyword((MK, $\mathcal{D}_P, id, w$), EDB)**:**

- Data owner: Take the master key MK $= (k_1, k_2, p, q)$, dictionary $\mathcal{D}_P$, encrypted database EDB $= (\mathcal{D}_W, \mathcal{D}_F,\ DT_{id})$ and a chosen file-keyword pair (id, w) as inputs, then execute **Add Keyword Algorithm** to add the new ciphertext of the pair to EDB.
- Server: Take $EDB = (\mathcal{D}_W, \mathcal{D}_F, \mathcal{D}_{F,W})$ and $(L_w, D_w, L_{id}, D_{id}, L_{id,w}, D_{id,w})$ as inputs, and then run standard data algorithm **Update**$(D_W, (L_w, D_w))$, and **Update**$(\mathcal{D}_{F,W}, (L_{id,w},\ D_{id,w}))$.

Algorithm 2. Add Keyword Algorithm

```
Require: MK, $\mathcal{D}_P, id, w$, EDB;
Ensure: $K_1, K_2, K_3$
 1: $P_w \leftarrow$ Get$(\mathcal{D}_p, w)$;
 2: if $P_w = NULL$ then
 3:     $L_w \leftarrow F_{k_1}(g^{1/w} \mod n)$
 4: else
 5:     $L_w \leftarrow P_w$
 6: end if
 7: $R_w \xleftarrow{\$} \{0,1\}^k$, $D_w \leftarrow H(F_{k_2}(g_1^{1/w} \mod n), R_w) \oplus (0||id||P_w)$
 8: $K_1 \leftarrow (L_w, D_w, R_w)$
 9: Update$(\mathcal{D}_p(w, P_w))$
10: $P_{id} \leftarrow$ Get$(\mathcal{D}_p, id)$
11: if $P_{id} = NULL$ then
12:     $L_{id} = F_{k_1}(g^{1/id} \mod n)$
13: else
14:     $L_{id} \leftarrow P_{id}$
15: end if
16: $R_{id} \xleftarrow{\$} \{0,1\}^{2k}, D_{id} \leftarrow G(F_{k_2}(g^{1/id} \mod n), R_{id}) \oplus (0||L_w||L_{id,w}||P_{id})$
17: $K_2 \leftarrow (L_{id}, D_{id}, R_{id})$
18: $L_{id,w} \leftarrow F_{k_1}(g^{\frac{1}{id \cdot w}} \mod n)$,$R_{id,w} \xleftarrow{\$} \{0,1\}^k$
19: $D_{id,w} \leftarrow H(F_{k_2}(g^{\frac{1}{id \cdot w}} \mod n), R_{id,w}) \oplus (0||L_w||L_{id})$
20: $K_3 \leftarrow (L_{id,w}, D_{id,w}, R_{id,w})$
21: return $K_1, K_2, K_3$
```

DeleteFile$((MK, id), EDB)$**:**

- Data owner: Take $K = (k_1, k_2, p, q)$, $\mathcal{D}_P$ and a file identifier id as inputs, generate and send a delete token

$$DT_{id} = (F_{k_1}(g^{1/id} \mod n), F_{k_2}(g^{1/id} \mod n))$$

  to the server.
- Server: Take the encrypted database $EDB = (\mathcal{D}_W, \mathcal{D}_F,\ DT_{id})$ as inputs, set $L_{id} = F_{k_1}(g^{1/id} \mod n)$, and executes the following algorithm to delete the expected file.

Algorithm 3. **Delete File Algorithm**

```
1: procedure DELETEFILE(((MK, id), EDB))
2:     $D_{id} \leftarrow \mathbf{Get}(\mathcal{D}_F, L_{id})$
3:     if $P_{id} = NULL$ then
4:         return $\perp$
5:     else
6:         $(D_{id,1}, D_{id,2}) \leftarrow D_{id}$
7:         $T||L_w||L_{id,w}||P_{id} = D_{id,1} \oplus G(F_{k_2}(g^{1/id} \mod n), D_{id,2})$
8:         $\mathbf{Remove}(\mathcal{D}_F, L_{id})$
9:         if T=0 then
10:            $D_w \leftarrow \mathbf{Get}(\mathcal{D}_w, L_w)$, $(D_{w,1}, D_{w,2}) \leftarrow D_w$
11:            $D_{w,1} = D_{w,1} \oplus (1||0^{2k})$
12:            $\mathbf{Update}(\mathcal{D}_w, (L_w, D_w(D_{w,1}, D_{w,2}))), \mathbf{Remove}(\mathcal{D}_F, L_{id})$
13:            $L_{id} \leftarrow P_{id}$
14:        end if
15:    end if
16: end procedure
```

Search$((sk_{\mathbf{w}}), \mathrm{EDB})$**:**

- Client: Whenever the client with searchable ability on keywords $\mathbf{w} = (w_1, w_2, \cdots, w_n)$ wants to search the file on keyword w_i, he uses his private key as inputs, compute the search token

$$ST_{w_i} = (F_{sk_{\mathbf{w},1}}(sk_{\mathbf{w},3}^{\prod_{w\in\mathbf{w}/\{w_i\}} w} \mod n), F_{sk_{\mathbf{w},2}}(sk_{\mathbf{w},3}^{\prod_{w\in\mathbf{w}/\{w_i\}} w} \mod n))$$

and send $ST_{w_i} = (F_{k_1}(g^{1/w_i} \mod n), F_{k_2}(g^{1/w_i} \mod n))$to the server;
- Server: Take EDB $= (\mathcal{D}_W, \mathcal{D}_F)$ and token $ST_{w_i} = (F_{k_1}(g^{1/w_i} \mod n))$, $F_{k_2}(g^{1/w_i} \mod n))$ as inputs, initialize an empty set $\mathcal{I}$, a temporary index-data pair $(L_w^t = NULL, D_w^t = NULL)$ and a temporary pointer $P_w^t = NULL$, set $L_w = F_{k_1}(g^{1/w_i})$, and do the following steps:

4 Security Analysis

In this section, we show that our proposed protocol is IND-CKA2 secure against the adaptive server and the client one after another as [24] except some leakage function. Before starting our proof, we need a simulator $\mathcal{S}$ to response the query from $\mathcal{A}$, which is defined in Sect. 2, to take the following leakage functions as input:

Theorem 1. *Suppose hash functions H and G and key-based pseudo-random function F_{k_1} are respectively modeled as three random oracles. Our complete DSSE scheme is IND-CKA2 secure with leakage functions in the random oracle model, where $(\mathcal{L}_{Setup} = |DB|$, $\mathcal{L}_{AddKeyword} = New(id, w)$, $\mathcal{L}_{DeleteFile} = (Old(id), New(id))$, $New(id, w))$ and $\mathcal{L}_{Search} = (DB(w), Old(w),\ New(w))$.*

Algorithm 4. Search Algorithm

```
Require: $(ST_{w_i}, \mathrm{EDB})$
Ensure: $\mathcal{I}$
 1: $D_w \leftarrow \mathbf{Get}(\mathcal{D}_W, L_w)$
 2: if $D_w = NULL$ then
 3:     return $\perp$
 4: else
 5:     $D_w = (D_{w,1}, D_{w,2})$
 6:     $T||id||P_W = D_{w,1} \oplus H(F_{k_2}(g^{\frac{1}{id \cdot w}} \mod n), D_{w,2})$
 7:     $L_w^t = L_w, D_w^t = D_w, P_w^t = P_w, L_w = P_w$
 8:     if T=0 then
 9:         $\mathcal{I} \leftarrow \mathcal{I} \cup id$, $L_w^t \leftarrow L_w, D_w^t \leftarrow D_w, P_w^t \leftarrow P_w$
10:     else
11:         if T=1 then
12:             $D_w^t \leftarrow (D_{w,1}^t, D_{w,2}^t)$, $D_{w,1}^t \leftarrow D_{w,1}^t \oplus (0^{k+1}||(P_w^t \oplus P_w))$
13:             $\mathbf{Update}(\mathcal{D}_w, (L_w^t, D_w^t = (D_{w,1}^t, D_{w,2}^t))$, $\mathbf{Remove}(\mathcal{D}_w, L_w)$
14:         end if
15:     end if
16: end if
17: return $\mathcal{I}$
```

The proof of Theorem 1 relies on Lemmas 1, 2, 3 and 4 defined below in [24], which just construct a map $f : x \rightarrow g^x$, here x can be w or id. Now it needs an efficient simulator $\mathcal{S}$ to play game $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(k)$ with an adversary $\mathcal{A}$. Our main arguments are the each lemma listed in following must be computational indistinguishable from the real one with leakage functions in the view of $\mathcal{A}$ under several complexity assumptions.

1. $\mathcal{L}_{Setup} = |DB|$: After running **Setup** algorithm, one will statistics the number of file-keyword pairs in DB according to the size of EDB.
2. $\mathcal{L}_{AddKeyword} = New(id, w)$: When running the **AddKeyword** algorithm, one will get the generated ciphertexts $New(id, w)$ by comparing with the former database.
3. $\mathcal{L}_{DeleteFile} = (Old(id), New(id))$: When deleting a selected file id, one will know all deleted ciphertexts of file which identifier is id, and ciphertexts of them were simulated by protocol **Setup** or **AddKeyword**.
4. $\mathcal{L}_{Search} = (DB(w), Old(w), New(w))$: When searching a file which contains the keyword w, one will know all matched ciphertexts and their father files in DB(w), and the first part of these ciphertexts were generated by protocol **Setup** or **AddKeyword**.

Lemma 1. *Suppose that there exists an adversary $\mathcal{A}$ that run protocol **Setup** to get the corresponding encrypted database from $\mathcal{S}$, and the leakage function $\mathcal{L}_{Setup} = |DB|$, then $\mathcal{A}$ can not distinguish the above simulated* EDB *with a real one.*

Lemma 2. *Suppose that H and G are random oracles, then for any polynomial time adversary $\mathcal{A}$, there exists an algorithm $\mathcal{S}_{AddKeyword}$, such that $\mathcal{A}$ could distinguish it with a real one that is generated in game $\boldsymbol{Real}_{\mathcal{A}}(k)$.*

Lemma 3. *Suppose H and F_{k_1} are random oracles, then for any polynomial time adversary $\mathcal{A}$, there exists an algorithm $\mathcal{S}_{Search}$, such that $\mathcal{A}$ could distinguish it with a real one that is generated in game $\boldsymbol{Real}_{\mathcal{A}}(k)$.*

Lemma 4. *Suppose G and F_{k_1} are random oracles, then for any polynomial time adversary $\mathcal{A}$, there exists an algorithm $\mathcal{S}_{DeleteFile}$, such that $\mathcal{A}$ could distinguish it with a real one that is generated in game $\boldsymbol{Real}_{\mathcal{A}}(k)$.*

We define algorithm **Setup**, **DeleteFile**, **AddKeyword**, **Search** be the event C_i for $i = 1, 2, \cdots, 4$ respectively. From the four lemmas above, we have that the distinguish probability of them each can be write as $|Pr[\mathbf{Real}_{\mathcal{A}}^{C_i}(k) = 1] - Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{C_i}(k) = 1]| \leq \epsilon_i$, where $1 \leq i \leq 4$, and ϵ_i are all negligible.

Summarily, the indistinguishability of above four protocols implies that $\mathcal{A}$ can not distinguish game $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(k)$ with game $\mathbf{Real}_{\mathcal{A}}(k)$. Because, we have that the probability $|Pr[\mathbf{Real}_{\mathcal{A}}(k) = 1] - Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(k) = 1]|$ is also negligible, which can be got by the computation below:

$$\begin{aligned} &|Pr[\mathbf{Real}_{\mathcal{A}}(k) = 1] - Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(k) = 1]| \\ &= \prod_{i=1}^{5} |Pr[\mathbf{Real}_{\mathcal{A}}^{C_i}(k) = 1] - Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{c_i}(k) = 1]| = \prod_{i=1}^{5} \epsilon_i \end{aligned}$$

This completes the proof of Theorem 1.

Theorem 2. *Our scheme Π is secure against malicious clients, i.e., search token in Π is unforgeable against adaptive attacks, assuming that the strong RSA assumption holds.*

Assume that there exists an adversarial client $\mathcal{A}$ who can generate a valid search token for some nonauthorized keyword w_0, so he can get the correct value $(g^{1/w'} \bmod n)$. In this case, we can use $\mathcal{A}$ to construct an efficient algorithm $\mathcal{B}$ to solve the strong RSA problem with a non-negligible probability by Euclidean algorithm. Consider the properties of RSA function, actually unless the client can compute the correct value $g^{1/w'} \bmod n$, or no one can generate a valid search token for non-authorized keyword w'.

5 Comparison and Analysis

In this section, we simply analyze the efficiency of our scheme by providing the cost of communication and computation in our scheme. Here we set all the number of keywords be one so to compare easily. Let $|G|, |\mathbb{Z}_p|$ respectively be the size of the group element $\mathbb{G}$, security parameter size. *exp* denotes the

Table 1. The communication and computation cost of some classical retrieval scheme

Scheme	Key size	Cipher. size	Search cost	Dynamic	Multi-client
Xu et al. [24]	$2\|k\|$	$O(\|DB\|)$	$O(\|DB(w)\|)$	$\surd$	
Sun et al. [25]	$3\|k\| + \|G\|$	$3O(\|DB(w)\|)$	$O(\|DB(w)\| \cdot exp)$		$\surd$
Ours	$2k + \|G\|$	$O(\|DB\|)$	$O(\|DB(w)\|)$	$\surd$	$\surd$

computation cost of the exponential operation. Table 1 lists some classical similar schemes about searchable encryption.

From the table, we can see that our searchable encryption achieves a balance in diversified function and communication cost. The size of EDB can keep the size of $O(|DB|)$, which is similar with the scheme proposed by Xu [24]. And we also realize the multi-client function in our paper without increasing much computation cost.

6 Conclusion

We construct an efficient and practical multi-client symmetric searchable encryption scheme with physical deletion property via RSA function in the random oracle model, and prove the security of the scheme by using the strong RSA function and four attack lemmas. The scheme gives a general method to extend the single reader model searchable encryption scheme to multiple readers. We also present the detailed communication cost and computation cost of the proposed scheme and point out that our scheme is more efficient than other classical ones by comparing the running time with some classical searchable encryption in each phase.

Acknowledgments. This work is partially supported by the Fundamental Research Funds for the Central Universities (No. 30916011328), the National Natural Science Foundation of China (61702342), and Shenzhen Science and Technology Program (JCYJ20170302151321095, JCYJ20170302145623566). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

1. Liu, J.K., Au, M.H., Susilo, W., et al.: Secure sharing and searching for real-time video data in mobile cloud. IEEE Netw. **29**(2), 46–50 (2015)
2. Baek, J., Vu, Q.H., Liu, J.K., et al.: A secure cloud computing based framework for big data information management of smart grid. IEEE Trans. Cloud Comput. **3**(2), 233–244 (2015)
3. Wang, S., Zhou, J., Liu, J.K., et al.: An efficient file hierarchy attribute-based encryption scheme in cloud computing. IEEE TIFS **11**(6), 1265–1277 (2016)
4. Wang, S., Liang, K., Liu, J.K., et al.: Attribute-based data sharing scheme revisited in cloud computing. IEEE TIFS **11**(8), 1661–1673 (2016)

5. Chor, B., Kushilevitz, E., Goldreich, O., et al.: Private information retrieval. J. ACM **45**, 965–981 (1998)
6. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24852-1_3
7. Liu, C., Zhu, L., Wang, M., et al.: Search pattern leakage in searchable encryption: attacks and new construction. Inf. Sci. **265**, 176–188 (2014)
8. Liu, J., Lai, J., Huang, X.: Dual trapdoor identity-based encryption with keyword search. Soft. Comput. **21**(10), 2599–2607 (2015)
9. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_20
10. Jarecki, S., Jutla, C., Krawczyk, H., et al.: Outsourced symmetric private information retrieval. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, pp. 875–888. ACM, Berlin (2013)
11. Liang, K., Huang, X., Guo, F., Liu, J.K.: Privacy-preserving and regular language search over encrypted cloud data. IEEE TIFS **11**(10), 2365–2376 (2016)
12. Kasra Kermanshahi, S., Liu, J.K., Steinfeld, R.: Multi-user cloud-based secure keyword search. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 227–247. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_12
13. Yang, X., Lee, T.T., Liu, J.K., et al.: Trust enhancement over range search for encrypted data. In: Trustcom, pp. 66–73. IEEE, New York (2016)
14. Zuo, C., Macindoe, J., Yang, S., et al.: Trusted Boolean search on cloud using searchable symmetric encryption. In: Trustcom, pp. 113–120. IEEE, New York (2016)
15. Liang, K., Su, C., Chen, J., Liu, J.K.: Efficient multi-function data sharing and searching mechanism for cloud-based encrypted data. In: Proceedings of the 11th ACM on Asia CCS, pp. 83–94. ACM (2016)
16. Curtmola, R., Garay, J., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: CCS 2006, pp. 79–88. ACM, New York (2006)
17. Bao, F., Deng, R.H., Ding, X., Yang, Y.: Private query on encrypted data in multi-user settings. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 71–85. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79104-1_6
18. Dong, C., Russello, G., Dulay, N.: Shared and searchable encrypted data for untrusted servers. In: Atluri, V. (ed.) DBSec 2008. LNCS, vol. 5094, pp. 127–143. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70567-3_10
19. Wang, Q., Zhu, Y., Luo, X.: Multi-user searchable encryption with fine-grained access control without key sharing. In: International Conference on Advanced Computer Science Applications and Technologies, pp. 119–125. IEEE (2014)
20. Wang, S., Zhang, X., Zhang, Y.: Efficiently multi-user searchable encryption scheme with attribute revocation and grant for cloud storage. PLoS ONE **11**(11), e0167157 (2016)
21. Wang, Y., Wang, J., Sun, S.-F., Liu, J.K., Susilo, W., Chen, X.: Towards multi-user searchable encryption supporting Boolean query and fast decryption. In: Okamoto, T., Yu, Y., Au, M.H., Li, Y. (eds.) ProvSec 2017. LNCS, vol. 10592, pp. 24–38. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68637-0_2

22. Cui, H., Deng, R.H., Liu, J.K., Li, Y.: Attribute-based encryption with expressive and authorized keyword search. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 106–126. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_6
23. Van Rompay, C., Molva, R., Önen, M.: Multi-user searchable encryption in the cloud. In: Lopez, J., Mitchell, C.J. (eds.) ISC 2015. LNCS, vol. 9290, pp. 299–316. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23318-5_17
24. Xu, P., Liang, S., Wang, W., Susilo, W., Wu, Q., Jin, H.: Dynamic searchable symmetric encryption with physical deletion and small leakage. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 207–226. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_11
25. Sun, S.-F., Liu, J.K., Sakzad, A., Steinfeld, R., Yuen, T.H.: An efficient non-interactive multi-client searchable encryption with support for Boolean queries. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 154–172. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_8