Enabling Dynamic Multi-Client and Boolean Query in Searchable Symmetric Encryption Scheme for Cloud Storage System

Wanshan Xu[1], Jianbiao Zhang[1*], and Yilin Yuan[1]
[1] Faculty of Information Technology, Beijing University of Technology
Beijing 100124 , China
[e-mail: xuwanshanxws@126.com]
[*]Corresponding author: Jianbiao Zhang

Received July 12, 2021; revised February 18, 2022; accepted March 18, 2022; published April 30, 2022

Abstract

Searchable symmetric encryption (SSE) provides a safe and effective solution for retrieving encrypted data on cloud servers. However, the existing SSE schemes mainly focus on single keyword search in single client, which is inefficient for multiple keywords and cannot meet the needs for multiple clients. Considering the above drawbacks, we propose a scheme enabling dynamic multi-client and Boolean query in searchable symmetric encryption for cloud storage system (DMC-SSE). DMC-SSE realizes the fine-grained access control of multi-client in SSE by attribute-based encryption (ABE) and novel access control list (ACL), and supports Boolean query of multiple keywords. In addition, DMC-SSE realizes the full dynamic update of client and file. Compared with the existing multi-client schemes, our scheme has the following advantages: 1) Dynamic. DMC-SSE not only supports the dynamic addition or deletion of multiple clients, but also realizes the dynamic update of files. 2) Non-interactivity. After being authorized, the client can query keywords without the help of the data owner and the data owner can dynamically update client's permissions without requiring the client to stay online. At last, the security analysis and experiments results demonstrate that our scheme is safe and efficient.

Keywords: searchable symmetric encryption, multi-client, Boolean query, attribute-based encryption, cloud storage.

This research was supported by Beijing Natural Science Foundation, the Beijing government [M21039].

http://doi.org/10.3837/tiis.2022.04.011 **ISSN : 1976-7277**

1. Introduction

The development of cloud computing has brought great convenience for the public, more and more users outsource their data to the cloud. The advantages of the cloud storage, such as mobile access, stability and reliability, make the users can access data anytime and anywhere, which greatly improves work efficiency and realize resource sharing while ensuring data security.

To ensure the confidentiality and integrity of cloud storage, the data is encrypted before it is uploaded to the cloud. But unfortunately, performing keyword search on ciphertext is a difficult task for the user. When searching for a particular protocol to achieve the user must download the cipher-text and decrypt it after searching. It is extremely inefficient and impractical when the scale of the data is very large. Therefore, searchable encryption (SE) ([1], [2], [3], [4]) came into being.

SE allows the user to search keywords on the ciphertext without revealing their privacy. SE performs queries on the ciphertext, and files to be searched is transparent to the server, which helps to achieve the integrity and confidentiality of the data on the cloud. And furthermore, it is conducive to protecting user privacy.

Searchable symmetric encryption (SSE) ([5], [6], [7], [8]) is an efficient and secure SE. Assisting with the inverted index and symmetric encryption primitives, the SSE achieves efficient ciphertext retrieval in sublinear time. Although SSE is an efficient means of ciphertext retrieval, but now most of the existing SSE schemes mainly focus on the search of a single-keyword in a single-client setting, which limits the expansion of SSE in practice. The multi-client SSE scheme was first proposed by Curtmola [9] in 2006, and then multi-client schemes were proposed one after another. However, the existing multi-client SSE schemes are mostly interactive (the client interacts with the data owner when performing keyword retrieval), and do not support the dynamic update of the client (dynamic addition and deletion of the client) or the dynamic update of files.

Related works. Searchable encryption was proposed by Song [1] in 2000, which is a full text search, the search cost grows linearly with the size of the database. To improve query efficiency, Curtmola [9] proposes a symmetric searchable encryption scheme with inverted index to achieve optimal search time. Chase and Kamara [10] propose a similar scheme but costs higher storage. In addition to search efficiency, many works have been done to improve query expression ([13], [14], [15], [16], [17]) and advanced security ([20], [24], [25], [26], [27], [28]).

The original SSE scheme was mainly for single-keyword, to enrich the search function, some research focus on multi- keyword SSE scheme. Golle [13] and Ballard [14] proposed efficient conjunctive keyword searches over encrypted data, these two schemes can realize multi-keyword queries, but the communication cost is linear in the number of documents. To provide a truly practical search capability, Cash [15] proposed a highly-scalable searchable symmetric encryption with support for Boolean queries, which constructs the OXT protocol to achieve Boolean query. Based on OXT protocol, Lai [16] proposed a result pattern hiding SSE scheme supporting conjunctive queries. Xu [17] proposed EGRQ, a range query scheme to achieve secure and efficient query on encrypted spatial data. The above SSE schemes support multi-keyword query, but only supports single-client scenarios. The concept of multi-client symmetric searchable encryption (MSSE) was first proposed by Curtmola [9], which uses broadcast encryption on top of a single-client scheme. Raykova [18] improves the efficiency of Curtmola by employing a deterministic encryption and achieves a linear search time. These two schemes are interactive and have a large communication cost. Jarecki [19] extends the

OXT scheme proposed by Cash [15] to the multi-client by the utilization of homomorphic signature and oblivious pseudorandom functions (PRFs), and realizes the Boolean query in the multi-client setting. Faber [21] extends the query type of OXT, supporting for range, substring, wildcard, and phrase queries. However, both these two schemes are interactive. Du [23] presented a multi-client SSE scheme that supports Boolean queries, which incorporates a client's authorization information into search tokens and indexes. The scheme proposed by Du supports dynamic update of client permissions, however, the data owner regenerates the search index every time a new client joins, which becomes very inefficient when the index scale is large.

Contributions. We propose a dynamic multi-client searchable symmetric encryption scheme supporting Boolean query, which extends SSE from the single-client setting to multiple clients, while realizing dynamic update of clients and fine-grained access control. In addition, our scheme supports Boolean query with multiple keywords. The main contributions are summarized as follows:

1. We use attribute-based encryption (ABE) to extend SSE from single client to multi-client, and implement Boolean query of multiple keywords. We construct a hybrid encryption that symmetric encryption is used to encrypt files and ABE is used to encrypt the symmetric key, only the client meeting the access policy can decrypt the key so as to decrypt files, an efficient SSE scheme in multi-client is implemented.
2. We implement efficient dynamic update (add/delete) of clients, and the time cost is $O(1)$ for N clients. By constructing the access control list (ACL), our scheme only allows authorized clients to access keywords, so as to prevent malicious clients from illegal access.
3. We have realized the dynamic update of files, and the data owner can update files independently without affecting other clients. Furthermore, in our scheme, the deleted files can be filtered by judging the operation (add/delete) when the server performs a search, and only valid files are sent back to the client, which improves the search efficiency and reduces the communication load.

2. Preliminaries

2.1 System model

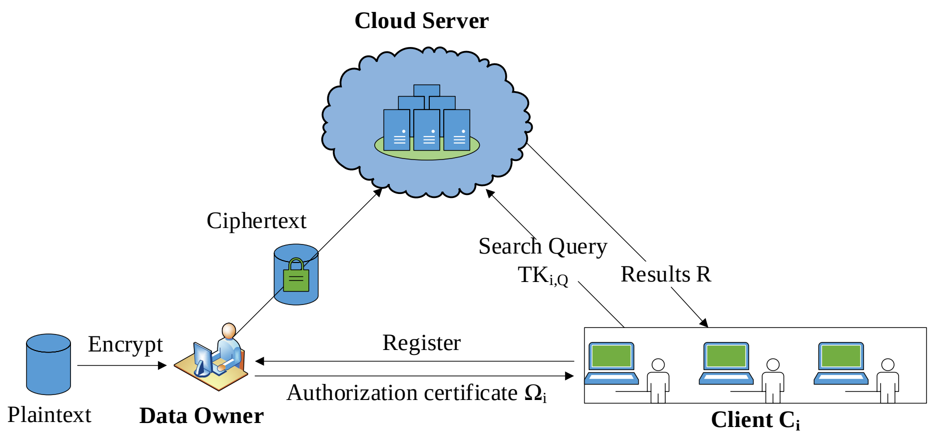


Fig. 1. System model

The system model of DMC-SSE is shown in **Fig. 1**, there are three entities in the system: data owner D, client C_i and the cloud sever. The server provides cloud storage services and is honest-but-curious, also it is not trusted. The data owner encrypts the plaintext DB (a database including a list of d identifier-keyword pairs $(\mathrm{id}_i, \mathrm{W}_{id=i})_{i=1}^{d}$) to ciphertext EDB, and sends it to the cloud. To perform a query Q with keywords ($\overline{W} = \{w_1, w_2, ..., w_n\}$) from the server, the client C_i needs to register to the data owner D first, and then D will returns an authorization certificate Ω_i to the client, with which the client C_i generates a search token $TK_{i,Q}$ and sends it to the server. On receiving $TK_{i,Q}$, the server will search the EDB and returns the results R that satisfies the requirement to the client C_i, finally C_i decrypts the files in R locally.

2.2 Attribute-based encryption (ABE)

Attribute-based encryption (ABE) is developed from the encryption scheme based on fuzzy identity, which can be divided into two types: key strategy ABE (KP-ABE) and ciphertext strategy ABE (CP-ABE). KP-ABE allows the private key to correspond to an access structure, and the ciphertext corresponds to an attribute set; while CP-ABE, on the contrary, allows the private key to correspond to an attribute set, and the ciphertext corresponds to an access structure. Whether KP-ABE or CP-ABE, only the attribute set satisfy the access policy can decrypt the ciphertext. ABE is very effective in encrypting data sharing, since it can realize data access control while encrypting data. In this paper, we use CP-ABE attribute encryption scheme, which contains the following four algorithms:

- $\mathrm{ABE.Setup}(\lambda)$: takes secret parameter λ as the input and outputs the system parameter mpk and master key msk .
- $\mathrm{ABE.KeyGen}(msk, mpk, \mathrm{A})$:takes the system parameter mpk ,master key msk and the attribute set A as the input and outputs a private key sk_A^i .
- $\mathrm{ABE.Enc}(mpk, msg, \mathrm{U})$:takes the message msg , system parameter mpk and the access structure U as the input and outputs the ciphertext $msg*$.
- $\mathrm{ABE.Dec}(sk_A^i, msg*)$: takes the ciphertext $msg*$ and the private key sk_A^i as the input, $msg*$ contains an access structure U and sk_A^i is associated with a set of attribute A , this algorithm outputs the decrypted information msg if $\mathrm{A} \in \mathrm{U}$.

3. Overview

The multi-client SSE scheme in our system combines attribute-based encryption with searchable symmetric encryption. First, files in DB are encrypted by the symmetric key K_c, and then K_c is encrypted by attribute-based encryption. Only clients who meet the attribute policy can decrypt the symmetric key and decrypt the ciphertext.

3.1 Access Control List (ACL)

The access control list (ACL) is owned by the data owner to control the permission of other clients. Assume that the client C_i will joins our system, it registers with the data owner D . To identify the client C_i , D generates a tag α_i for C_i , $\alpha_i \in \mathrm{Z}_p^*$. For legitimate keywords $(w_1, w_2, ..., w_n)$ that can be accessed by C_i, D computes $\mathrm{c}_j \leftarrow \alpha_i \cdot w_j$, $j = 1, 2, ..., n$, and adds c_j to

ACL, at last, D sends the updated ACL to the cloud server.

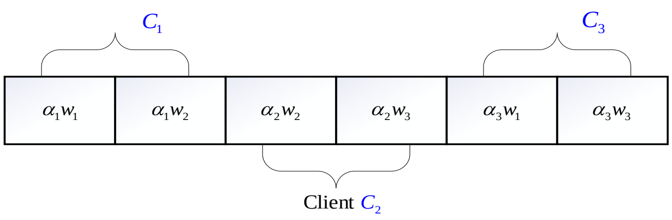


Fig. 2. The structure of ACL

Assume that there are three clients ($\boldsymbol{C}_1$, $\boldsymbol{C}_2$, $\boldsymbol{C}_3$) with the tag ($\alpha_1,\alpha_2,\alpha_3$) and the corresponding authorization keywords are (w_1, w_2), (w_2, w_3) and (w_1, w_3),the structure of ACL is shown in **Fig. 2**.

When the client performs a query, it calculates $\mathrm{c_i}$ as above, and the server checks whether $\mathrm{c_i}$ is in ACL, if so, the query is legal and can be continued, else the query is illegal and the query will be stopped.

$$\mathrm{c}_i \begin{cases} \mathrm{c}_i \in ACL, \text{query is legal,continue} \\ \mathrm{c}_i \notin ACL, \text{query is illegal,stop} \end{cases}$$

The structure of the ACL is a one-way list, and the ACL is updated dynamically according to the change of client's search permission. When a new client joins, the c_j of the client are added at the tail of the ACL while the values will be removed when the client is revoked.

3.2 Scheme Definition

Our scheme mainly includes the following algorithms：

- KeyGen(1^λ): takes the system parameter λ as input, and outputs the system master key msk and public key mpk. It is performed by the data owner D .
- EDBSetup(DB, msk, mpk, U): takes the database DB, system master key msk, master public key mpk and an attribute universe U as input, and outputs the encrypted database EDB.
- ClientAuth(msk, mpk, A_i ,ACL): the client C_i submits its attribute A_i to the data owner D , D generates the private key sk_A^i of C_i with msk , mpk and C_i 's attributes A_i , meanwhile, D assigns the client C_i an identity α_i ,which will be encrypted by $\mathrm{enr}_i \leftarrow \mathrm{ABE.Enc}(\alpha_i)$,the sk_A^i and enr_i will be sent back to the client. In addition, D will calculates $\mathrm{c}_i \leftarrow \alpha_i \cdot w_i$ for legal keywords of client and adds c_i to ACL. At last, D sends the updated ACL to the server.
- TokenGen(sk_A^i , enr_i , W): the client C_i takes private key sk_A^i , encrypted identity enr_i and keywords W to query as input, and generates the token $TK_{i,Q}$ as output.
- Search ($TK_{i,Q}$, EDB): the sever takes the token $TK_{i,Q}$ as input, and outputs results R that satisfy the query requirements to the client.
- Retrieve(sk_A^i , R):the client C_i gets the identifiers of documents by decrypting the

returned results R with her private key sk_A^i .With the identifiers and decryption key , the client will retrieve the original documents.

3.3 Security Definition

In our scheme, we consider the security against adversarial server, it's the design goal of our scheme to reveal as little information as possible to the server in a query. The less information leaked to the server, the more difficult it is for the server to guess the information of the token or file, so as to better protect the privacy of users.

Security against adversarial server. A loss function L is used to represent the information leaked to the adversary during a query, let $\prod$ = {KeyGen, EDBSetup, ClientAuth, TokenGen, Search} be our DMC-SSE scheme, we define the security of $\prod$ by two experiments: $\mathrm{Real}_{\mathsf{A}}^{\prod}(\lambda)$ and $\mathrm{Ideal}_{\mathsf{A},\mathsf{S}}^{\prod}(\lambda)$:

$\mathrm{Real}_{\mathsf{A}}^{\prod}(\lambda)$: Then adversary A chooses a series of queries adaptively and repeatedly to trigger the experiment runs KeyGen, ClientAuth, TokenGen and Search and the experiment outputs a bit b that A returns to the experiment.

$\mathrm{Ideal}_{\mathsf{A},\mathsf{S}}^{\prod}(\lambda)$: Adversary A chooses a database DB and a series of queries Q ,the experiment runs $\mathsf{S}(\mathsf{L}(\mathrm{DB},\mathrm{Q}))$ and output a bit b .

Definition 1. We say that $\prod$ is L -semantically-secure if for any probabilistic, polynomial-time (PPT) adversaries A ,there exists a PPT simulator S , such that :

$$|\Pr[\mathrm{Real}_{\mathsf{A}}^{\prod}(\lambda)=1]-\Pr[\mathrm{Ideal}_{\mathsf{A},\mathsf{S}}^{\prod}(\lambda)=1]|\le negl(\lambda)$$

Now we describe the loss function L in our scheme. As ref [23], for simplicity of analysis, we consider a simple setting that all queries are conjunctive queries. We use $\mathrm{Q}=(\mathrm{s},\mathrm{x})$ to denote a series of conjunctive queries, where $Q[k] = (s[k], x[k,1], x[k,2],..., x[k,n])$ is an individual query, $s[k]$ and $x[k,\cdot]$ denote sterm (the least frequent keyword among all keywords in a query) and xterm (the other keywords except sterm in a query), respectively. The leakage function L (DB, Q) can be defined as below:

- $N=\Sigma_{i=1}^{d}|\mathrm{W}_i|$, the number of the (w_i, id_i) pairs.
- $\bar{\mathrm{s}}\in \mathsf{N}^T$, the equality pattern of sterms s, indicating which queries have the same sterm.
- SN , the number of files matching the sterm, obviously, $SN[k]=|\mathrm{DB}[k]|$.
- AN , the number of files matching the entire conjunction query, $\mathrm{AN}[k]=\mathrm{DB}(\mathrm{s}[k])\cap \mathrm{DB}(\mathrm{x}[k,\alpha])$, $\alpha=\{1,2,...,n\}$
- IP is the conditional intersection pattern, which is formally defined by
$$\mathrm{IP}[k_1,k_2,\alpha,\beta]=\begin{cases}\mathrm{DB}(\mathrm{s}[k_1])\cap\mathrm{DB}(\mathrm{s}[k_2]) & \text{if } k_1\ne k_2 \text{ and } \mathrm{x}[k_1,\alpha]=\mathrm{x}[k_2,\beta]\\ \phi & \text{otherwise}\end{cases}$$
- DBT is the search result pattern of the sterm in the k -th query, $\mathrm{DBT}[k]=\mathrm{DB}[\mathrm{s}[k]]$
- XN is the number of xterms in the k -th query.

4. Dynamic multi-client SSE

In this section, we give our multi-client searchable symmetric encryption scheme $\prod$ = { KeyGen, EDBSetup, ClientAuth, TokenGen, Search }. Let $H_i:\{0,1\}^*\rightarrow\{0,1\}^\lambda$, be hash functions, and $F:\{0,1\}^\lambda\times\{0,1\}^\lambda\rightarrow\{0,1\}^\lambda$ be PRFs.

4.1 Our construction

$\mathrm{KeyGen}(1^\lambda)$: with the system parameter λ,the data owner generates the master key msk and public key mpk, where $(mpk, msk) \leftarrow \mathrm{ABE.Setup}(1^\lambda)$.

$\mathrm{EDBSetup}(\mathrm{DB}, mpk, \mathsf{U})$: As shown in algorithm 1, data owner takes the database DB= $(id_i, W_i)_{i=1}^{d}$, public key mpk and an attribute set U as input, and outputs the encrypted database $\mathrm{EDB} = (\mathrm{T}, \mathrm{X})$. It chooses big primes p, q, random keys K_I, K_z for a PRF F_p and K_w for a PRF F, $g \xleftarrow{\$} \mathsf{G}$. To improve the efficiency of DB encryption and decryption, symmetric encryption primitives are used in our scheme. To share the symmetric key K_{id} with legitimate users, D encrypts K_{id} with the attribute set U, op represents the operation (add/delete) of the files.

EDB consists of a TSet T and a XSet X, these two sets are stored in dictionary structure, EDB uses inverted index to store the identifiers of all documents. Like most other MSSE schemes, identifiers of files in DB is encrypted and stored in T.

Algorithm 1 EDBSetup

Input: DB, mpk, U
Output: EDB

```
function EDBSetup (DB, mpk, U)
  T ← {}; X ← {}; cnt ← 0
  for w ∈ W do
    cnt ← 1; stag_w ← F(K_w, w)
    for id ∈ DB[w] do
      l ← H(stag_w || count); u ← ABE.Enc(mpk, id || K_id, U)
      eid ← F_p(K_I, id); z ← F_p(K_z, w)
      v ← eid · z^{-1}; o ← op ⊕ l
      x ← g^{F_p(K_x, w)·eid}; X ← X ∪ x;
      T[l] ← (u, v, o); cnt ← cnt + 1
    end for
  end for
  EDB ← {T,X}
end function
```

Algorithm 2 ClientAuth

Input: msk, mpk, ACL
Output: Ω ,ACL'

```
function ClientAuth (msk, mpk, ACL)
  sk_A^i ← ABE.KeyGen(msk, mpk, A_i)
  r_i ← {0,1}^λ; α_i ← F(K_k, r_i)
  enr ← ABE.Enc(α_i)
  for w ∈ W̄ do
    c_i ← α_i · w; ACL ← ACL ∪ c_i
  end for
  ACL' ← ACL; Ω_i ← {sk_A^i, K_w, K_z, K_x, mpk, enr}
```

return Ω_i
end function

ClientAuth(msk, mpk, A_i, ACL): When a client with attribute set A_i performs a query on the encrypted database for the first time, he needs to authenticate with the data owner. The data owner D generates a corresponding private key sk_A^i according to the properties A_i of the client C_i ,where $sk_A^i \leftarrow \text{ABE.KeyGen}(msk, mpk, \mathsf{A}_i)$, D sends the private key sk_A^i to the client. To ensure that legitimate clients can only access the authorized keywords, the data owner D first generates an identity α_i for client C_i , then uses α_i and legal keywords w to generate a blind factor c_i which will be added to ACL , keywords only that in ACL can be accessed by the client. At last, D sends the $\Omega_i \leftarrow (sk_A^i, K_w, K_z, K_x, mpk, \text{enr})$ back to the client C_i ,where $\text{enr} \leftarrow \text{ABE.Enc}(\alpha_i)$ and send the updated access control list ACL' to the server.

Algorithm 3 TokenGen
Input: Ω_i , Q= { $w_1, w_2, ..., w_n$ }
Output: $TK_{i,Q}$
function TokenGen (Ω_i , Q)
  $\text{acf}_w \leftarrow F(K_w, w_1)$; $\alpha_i \leftarrow \text{ABE.Dec(enr)}$
  for $c = 1, 2, ...$ until the server stops do
    for $j = 2, ..., n$ do
      $\text{xtoken}[c, j] \leftarrow g^{F_p(K_z, w_1 \| c) \cdot F_p(K_x, w_j)}$; $\text{ctl}[j] \leftarrow \alpha_i \cdot \text{w}_j$
    end for
  $TK_{i,Q} \leftarrow \{\text{acf}_w, \text{xtoken}[2], ..., \text{xtoken}[n], \text{ctl}[2], ..., \text{ctl}[n]\}$
  end for
  return $TK_{i,Q}$
end function

TokenGen(Ω_i, Q) :When the client C_i wants to perform a boolean search on the EDB with a set of keyword $\overline{w} = \{w_1, w_2, ..., w_n\}$, he first choose a sterm who is the keyword with lowest-frequency from $\overline{w}$, for simplicity, we assume that w_1 is the sterm and assume that we take the conjunctive query $Q = \{w_1 \wedge w_2 \wedge ... \wedge w_n\}$. The client C_i generates a blind factor $\text{ctl}[i] \leftarrow \alpha_i \cdot \text{w}_i$ for each keyword w_i ($i = 2, ..., n$) in xterms, where $\alpha_i \leftarrow \text{ABE.Enc(enr)}$, and the token will be generated by the algorithm 3.

Algorithm 4 Search
Input: TK_Q, EDB, ACL
Output: R
function Search (TK_Q, EDB, ACL)
  $\text{R} \leftarrow \{\}$; $c \leftarrow 0$
  while true do
    $\mathsf{l} \leftarrow H(\text{acf}_w \| c)$
    if T[l]=*null* then
      return R
    else

```
        (u, v, o) ← T[l] ; op ← u ⊕ l
        if op = "add" then
            if xtoken[i]^v ∈ X and ctl[i] ∈ ACL , i = 2,...,n then
                R ← R ∪ e
        else if op = "del" then
                R ← R - e
        c ← c + 1
      end if
    end while
end function
```

Search ($TK_{i,Q}$ EDB, ACL): On receiving the search token $TK_{i,Q}$ sent by the client C_i, the server will perform a search in EDB to find the matching files for $TK_{i,Q}$, and returns the file set R to the C_i, as is shown in algorithm 4.

To ensure the legitimate access of a query Q, the blind factor $\mathrm{ctl}[i]$ of the client is checked during the search, and only the keywords that the blind factor in the ACL are allowed to be accessed. Furthermore, only data owner D can perform op (add or delete) in function $\mathrm{EDBSetup}$, and other clients can only query keywords. If $op = "add"$, it indicates that the file is added and the corresponding id is valid, but for $op = "del"$, it indicates that the file is deleted, and the corresponding id is invalid.

After the client C_i retrieves the ciphertext R, it decrypts the symmetric key K_{id} with the algorithm $K_{id} \leftarrow \mathrm{ABE.Dec}(sk_A^i, e)$. Due to K_{id} is encrypted by attributes, only clients that satisfy the attribute encryption policy can decrypt it. With the symmetric key K_{id}, C_i can decrypt files efficiently.

4.2 Dynamic Update of Clients

In the multi-client SSE scheme, the dynamic update of the client is worth considering, since in practice, new clients may join the system at any time, and the clients in the system may be revoked at any time, too. In the scheme proposed by Du [23], when there is a client, the data owner D not only needs to update ACL, but also needs to update the encrypted database and regenerate the encrypted index according to the pk of the client, time cost is $O(n)$ for n clients, which is obviously inefficient when clients update frequently. However, in our scheme, due to the use of ABE, only the client whose attributes satisfy the access control policy can decrypt the ciphertext, therefore, the access rights of the file rely on the attributes of the client, and have nothing to do with the pk of the client, so the search index does not need to be regenerated no matter how many clients are added, the only thing that D has to do is updating the ACL, so time cost is $O(1)$ for n clients. The process of client revocation is similar, the difference is that the ACL update changes from adding to deleting. Therefore, our scheme is more efficient than Du [23] when clients update dynamically.

4.3 Dynamic Update of Files

In addition to the dynamic update of the client, another problem worth considering is the dynamic update of files. Because the files stored in the cloud are not immutable, the data owner may add new files or delete expired files, so the dynamic update of files is also necessary. We construct a novel dynamic operator op that denotes “add” or “delete” operation of files and op is encrypted together with the identifier when the data owner D updates the file. For query on the EDB, the server checks op and filters the documents whose $op = \text{"delete"}$, thus only valid files are reserved. The general SSE scheme returns all the matching files found, but there are some invalid files that are expired or to be deleted. In our scheme, the data owner D filters these invalid files through dynamic update of files, so that only valid files are returned, which reduces the communication load and improves the communication efficiency.

4.4 Supporting for Boolean queries

Given a query Q with keywords ($w_1, w_2, ..., w_n$), to support Boolean query, we use Boolean formula φ to construct the searchable form : $w_1 \wedge \varphi(w_2, ..., w_n)$. To perform a Boolean query, the client C_i sends the token $TK_{i,Q}$ to the server along with the Boolean formula φ . It’s the same with the algorithm 4 except that for a tuple (u, v, o) , instead of using $\text{xtoken}[i]^v \in X$, the server gets a binary value bv_i for each keywords in $(w_2, w_3, ..., w_n)$, where $bv_i = 1$ if the $\text{xtoken}[i]^v \in X$ corresponding to the keyword w_i , else $bv_i = 1$. After getting all the binary values, the server calculates the expression φ based on the values of $bv_2, bv_3, ..., bv_n$ and forwards u to R if the result is true.

4.5 Security Analysis

Theorem 1. Our scheme $\prod$ is L -semantically secure against adaptive attacks, where L is the leakage function defined in Definition 1, assuming that the DDH (Decisional Diffie-Hellman) assumption holds in G , F and F_p are secure PRFs and that $\sum = (\text{Enc,Dec})$ is an IND-CPA scheme.

Algorithm 5 G_0, $\boxed{G_1}$

function INITIALIZE (ACL, DB, s, x)
  $p, q, msk, mpk, g \xleftarrow{\$} \mathsf{Z}_n^*$ $K_w, K_I, K_z, K_x \xleftarrow{\$} \{0,1\}^\lambda$
  $\mathrm{T} \leftarrow \{\}; \mathrm{X} \leftarrow \{\}; cnt \leftarrow 0$; $(\mathrm{id}_i, \mathrm{W}_i) \leftarrow \mathrm{DB}$
  for $w \in W$ do
    $(\mathrm{id}_1, ..., \mathrm{id}_{\mathrm{T}_w}) \leftarrow \mathrm{DB}[\mathrm{w}]$; $\sigma \xleftarrow{\$} \mathrm{Perm}([\mathrm{T}_\mathrm{w}])$
    $\mathrm{WPerms}[\mathrm{w}] \leftarrow \sigma$; $\mathrm{stag}_\mathrm{w} \leftarrow F(K_w, w)$
    $\boxed{\mathrm{stags}[\mathrm{w}] \leftarrow \mathrm{stag}_\mathrm{w}}$
    for $cnt \in [T_w]$ do
      $\mathsf{l} \leftarrow H(\mathrm{stag}_\mathrm{w} \| cnt)$
      $u \leftarrow \mathrm{ABE.Enc}(mpk, \mathrm{id}_{\sigma[cnt]} \| K_{\mathrm{id}_{\sigma[cnt]}}, \mathsf{U})$
      $eid \leftarrow F_p(K_I, id_{\sigma[cnt]})$; $z \leftarrow F_p(K_z, w \| cnt)$
      $v \leftarrow eid \cdot z^{-1}$; $o \leftarrow op \oplus \mathsf{l}$
      $\mathrm{T}[\mathsf{l}] \leftarrow (u, v, o)$
    end for
  end for
  $\mathrm{X} \leftarrow \mathrm{XSETSETUP}(p, q, \mathrm{K_x}, \mathrm{K_I}, \mathrm{DB})$
  $\Omega_i \leftarrow \mathrm{CLIENTAUTH}(p, q, \mathrm{K}_k, \mathrm{ACL}, \overline{\mathrm{W}})$
  $\mathrm{EDB} \leftarrow \{\mathrm{T}, \mathrm{X}\}$
  for $k \in [T]$ do
    $\mathrm{t}[k] \leftarrow (\mathrm{EDB}, \Omega_\mathrm{i}, \mathrm{s}[k], \mathrm{x}[k,\cdot], \mathrm{c}[k,\cdot])$
  end for
    $eid \leftarrow F_p(K_I, id)$; $x \leftarrow g^{F_p(K_x, w) \cdot eid}$
    end for
  end function

function $\mathrm{CLIENTAUTH}(p, q, \mathrm{K}_k, \mathrm{ACL}, \overline{\mathrm{W}})$
  $\mathrm{r_i} \leftarrow \{0,1\}^\lambda$; $\alpha_\mathrm{i} \leftarrow F(K_k, \mathrm{r}_i)$
  $\mathrm{enr} \leftarrow \mathrm{ABE.Enc}(\alpha_\mathrm{i})$; $\boxed{\mathrm{A}[i] \leftarrow \alpha_\mathrm{i}}$
  for $\mathrm{w} \in \overline{\mathrm{W}}$ do
    $\mathrm{c}_j \leftarrow \alpha_\mathrm{i} \cdot \mathrm{w}$; $\mathrm{ACL} \leftarrow \mathrm{ACL} \cup \mathrm{c}_j$
  $\mathrm{KAL}' \leftarrow \mathrm{KAL}$; $\Omega_i \leftarrow \{K_w, K_z, K_x, \mathrm{enr}\}$
  return Ω_i
end function

function TRANSGEN ($\mathrm{EDB}, \Omega_i, \mathrm{s}[k], \mathrm{x}[k,\cdot]$
  $\alpha_i \leftarrow \mathrm{ABE.Dec(enr)}$; $\boxed{\alpha_\mathrm{i} \leftarrow \mathrm{A}[i]}$
  $\mathrm{acf} \leftarrow F(K_w, \mathrm{s}[k])$; $\boxed{\mathrm{acf} \leftarrow \mathrm{stags}[\mathrm{s}_k]}$
  for $\beta \in [|\mathrm{x}_k|]$ do
    for $cnt \in T$ do
      $\mathrm{xtoken}[\beta, cnt] \leftarrow \mathrm{g}^{F_p(K_z, \mathrm{s[k]} \| cnt) \cdot F_p(K_x, \mathrm{x[k},\beta])}$
    end for
    $\mathrm{ctl}[\beta] \leftarrow \alpha_i \cdot \mathrm{x}[\mathrm{k},\beta]$

end function	end for
	(stag,Res) $\leftarrow$ SEARCH(EDB,X,acf,xtoken,ctl)
function XSETSETUP($p,q,\mathrm{K_x},\mathrm{K_I}$,DB)	ResInds $\leftarrow \mathrm{DB}(\mathrm{s}[k]) \cap \mathrm{DB}(\mathrm{x}[k,1]) \cap \mathrm{DB}(\mathrm{c}[k,1])$
$(\mathrm{id}_i,\mathrm{W}_i) \leftarrow$ DB ; X $\leftarrow \phi$	$\ldots \cap \mathrm{DB}(\mathrm{x}[k,\mathrm{n}]) \cap \mathrm{DB}(\mathrm{c}[k,\mathrm{n}])$
for $w \in \mathrm{W}$ and ind $\in \mathrm{DB}(w)$ do	return (stag,xtoken,Res,ResInds)
X $\leftarrow \mathrm{X} \cup x$	end function

Proof: The proof can be conducted by constructing a sequence of games. Among these games, G_0 is designed to have the same distribution as $\mathrm{Real}^{\Pi}_{\mathrm{A}}(\lambda)$ and the last game G_8 is designed to simulate easily for the simulator S . In the proof of Theorem 1, the indistinguishability of the distribution between games proves that the simulator S satisfies the Definition 1, and the proof of the theorem is completed.

Game G_0 . As is shown in algorithm 5, G_0 is the real game with minor modifications for easy analysis. It takes (ACL, DB, s, x) as input to simulate EDBSetup in algorithm 1 by using function INITIALIZE. INITIALIZE is identical to EDBSetup except that X is separated as a subfunction, XSetup .

G_0 generates the transcript by using the function TransGen , before that, G_0 generates the secret key Ω_i by running function ClientAuth that simulates the ClientAuth algorithm as defined in algorithm 2, specifically, $\overline{\mathrm{W}}$ is the set of the authorized keywords, and the order of keywords are recorded in WPerms . For $k \in [T]$, G_0 runs function TransGen ($\mathrm{EDB},\Omega_{\mathrm{i}},\mathrm{s}[k],\mathrm{x}[k,\cdot],\mathrm{c}[k,\cdot]$) to output transcript $\mathrm{t}[k]$,the transcript is similarly as in the real game except the generation of ResInds : it gets ResInds by calculating $\mathrm{DB}(\mathrm{s}[k]) \cap \mathrm{DB}(\mathrm{x}[k,1]) \cap \mathrm{DB}(\mathrm{c}[k,1]) \ldots \cap \mathrm{DB}(\mathrm{x}[k,\mathrm{n}]) \cap \mathrm{DB}(\mathrm{c}[k,\mathrm{n}])$. G_0 has the same distribution with $\mathrm{Real}^{\Pi}_{A}(\lambda)$ assuming that no false positives happening, it's easy to get:

$$\Pr[G_0 = 1] - \Pr[\mathrm{Real}^{\Pi}_{adv} = 1] \le negl(\lambda)$$

Game G_1 . G_1 is identical to G_0 except the calculation of stag and α , the difference between G_1 and G_0 is shown in the boxed codes in algorithm 5. The values of stag and α will be recorded after being computed for the first time, and will be directly looked up instead of being computed again when used later. So we can get:

$$\Pr[\mathrm{G}_1 = 1] = \Pr[\mathrm{G}_0 = 1]$$

Algorithm 6 G₂, $\boxed{\mathrm{G_3}}$	
function INITIALIZE (DB, ACL, s, x, c, U)	**function** XSETSETUP(p,q,f_X,f_I,DB)
$p,q,msk,mpk,g \xleftarrow{\$} \mathbb{Z}^*_n$	$(\mathrm{id}_i,\mathrm{W}_i) \leftarrow$ DB ; X $\leftarrow \phi$
$K_w,K_I,K_z,K_x \xleftarrow{\$} \{0,1\}^{\lambda}$	for $w \in \mathrm{W}$ and ind $\in \mathrm{DB}(w)$ do
T $\leftarrow \{\}$;X $\leftarrow \{\}$;$cnt \leftarrow 0$; $(\mathrm{id}_i,\mathrm{W}_i) \leftarrow$ DB	$eid \leftarrow f_I(id)$; $x \leftarrow g^{f_X(w)\cdot eid}$
for $w \in W$ do	X $\leftarrow \mathrm{X} \cup x$
$(\mathrm{id}_1,\ldots,\mathrm{id}_{\mathrm{T_w}}) \leftarrow \mathrm{DB[w]}$; $\sigma \xleftarrow{\$} \mathrm{Perm}([\mathrm{T_w}])$	end for
$\mathrm{WPerms[w]} \leftarrow \sigma$; $\mathrm{stag_w} \leftarrow F(K_w,w)$	
$\mathrm{stags[w]} \leftarrow \mathrm{stag_w}$	end function
for $cnt \in [T_w]$ do	
$\mathsf{l} \leftarrow H(\mathrm{stag_w} \,\Vert\, cnt)$	**function TRANSGEN (** $\mathrm{EDB},\mathrm{A},f_z,f_X,\mathrm{s}[k],\mathrm{x}[k,\cdot]$ **)**
$u \leftarrow \mathrm{ABE.Enc}(mpk,\mathrm{id}_{\sigma[cnt]} \,\Vert\, K_{\mathrm{id}_{\sigma[cnt]}},\mathrm{U})$	
$\boxed{u \leftarrow \mathrm{ABE.Enc}(mpk,0^{\lambda},\mathrm{U})}$	$\alpha_i \leftarrow \mathrm{A}[i]$; acf $\leftarrow \mathrm{stags[s}_k]$
$eid \leftarrow f_I(id_{\sigma[cnt]})$; $z \leftarrow f_z(w \,\Vert\, cnt)$	for $\beta \in [\lvert \mathrm{x}_k \rvert]$ do
$v \leftarrow eid \cdot z^{-1}$; $o \leftarrow op \oplus \mathsf{l}$	for $cnt \in T$ do
$\mathrm{T}[\mathsf{l}] \leftarrow (u,v,o)$	$\mathrm{xtoken}[\beta,cnt] \leftarrow \mathrm{g}^{f_Z(\mathrm{s[k]}\Vert cnt)\cdot f_X(\mathrm{x[k},\beta])}$

end for
end for
$X \leftarrow \text{XSETSETUP}(p, q, f_X, f_I, \text{DB})$
$\Omega_i \leftarrow \text{CLIENTAUTH}(p, q, \text{K}_k, \text{ACL}, \overline{\text{W}})$
$\text{EDB} \leftarrow \{\text{T}, \text{X}\}$

for $k \in [T]$ do
  $\text{t}[k] \leftarrow (\text{EDB}, \text{A}, f_z, f_X, \text{s}[k], \text{x}[k, \cdot])$
end for
end function

end for
$\text{ctl}[\beta] \leftarrow \alpha_i \cdot \text{x}[\text{k}, \beta]$
end for
$(\text{stag}, \text{Res}) \leftarrow \text{SEARCH}(\text{EDB}, \text{X}, \text{acf}, \text{xtoken}, \text{ctl})$
$\text{ResInds} \leftarrow \text{DB}(\text{s}[k]) \cap \text{DB}(\text{x}[k,1]) \cap \text{DB}(\text{c}[k,1])$
$\quad \ldots \cap \text{DB}(\text{x}[k,\text{n}]) \cap \text{DB}(\text{c}[k,\text{n}])$
return (stag,xtoken,Res,ResInds)

end function

Game G_2 .The difference between G_2 and G_1 is that G_2 uses random functions instead of PRFs F and F_p, the details are shown in algorithm 6. Since $F(K_w, \cdot)$ and $F(K_k, \cdot)$ are calculated once for the same input, so we can replace them with random strings. $F_p(K_I, \cdot)$, $F_p(K_z, \cdot)$ and $F_p(K_x, \cdot)$ are replaced by f_I , f_z , f_x ,respectively. Note that, TransGen takes A as input so that CLIENTAUTH can be omitted. We can get that there exist adversaries $\mathsf{B}_{1,1}$ and $\mathsf{B}_{1,2}$ such that:

$$\Pr[\text{G}_2 = 1] - \Pr[\text{G}_1 = 1] \leq 2\text{Adv}_{F,\mathsf{B}_{1,1}}^{PRF}(\lambda) + 3\text{Adv}_{F,\mathsf{B}_{1,2}}^{PRF}(\lambda)$$

Game G_3 . G_3 is same as G_2 except for the code in the box, the details are shown in algorithm 6. G_3 uses an encryption of the constant string 0^λ to replace the encryption of file identifiers. Since the encryption operation is executed m times, so we can get that there exists an adversary B_2 which satisfies:

$$\Pr[\text{G}_3 = 1] - \Pr[\text{G}_2 = 1] \leq m \cdot \text{Adv}_{\Sigma,\mathsf{B2}}^{\text{ind-cpa}}(\lambda)$$

Game G_4 **.** As shown in algorithm 7, G_4 is same as G_3 except the way of generating X and xtoken. Different from G_3 , in G_4 , elements $\text{X\_Elem} = g^{f_X(w) \cdot f_I(id)}$ in X are precomputed and recorded in array $\text{H}(id, w)$ though the keyword w and the corresponding id . In G_4 , elements in X are generated in such a way: for a given $\text{w} \in \text{W}$ and $id \in \text{DB(w)}$, G_4 adds the value $\text{H}(id, w)$ form the array H to the set X. Recall that the value added to X is calculated by $g^{f_X(w) \cdot f_I(id)}$, which is same as G_3 .

As for the value in xtoken, in G_3 ,the xtoken is computed as $g^{f_Z(\text{s}_{\text{t}|} \| cnt) \cdot f_X(\text{x}[\text{k},\beta])}$, in G_4 , TransGen looks up $(i\text{d}_1, \ldots, \text{id}_{\text{T}_\text{s}}) \leftarrow \text{DB}(s_k)$, $\sigma \leftarrow \text{WPerms}[\text{s}_k]$ and $y \leftarrow f_I[\text{id}_{\sigma[\text{cnt}]}] \cdot f_Z[s \| cnt]$,the $\text{xtoken}[\beta]$ is set $H[id_{\sigma[cnt]}, \text{x}[\text{t},\beta]]^{1/v} = g^{f_X(\text{x}[\text{k},\beta]) \cdot f_Z(\text{s}_\text{t} \| cnt)}$,which is same as in G_3 . It's easy to see that:

$$\Pr[G_4 = 1] = \Pr[G_3 = 1]$$

Game G_5 . G_5 and G_4 are almost the same except the code in box in algorithm 7. Simplify, G_5 selects v form Z_p^* randomly instead of computing it. It is easy to see that:

$$\Pr[G_5 = 1] = \Pr[G_4 = 1]$$

Game G_6 . G_6 is almost identical to G_5 ,the difference is that instead of computing values of H and Y as the previous game G_5 , G_6 selects them form G randomly, the details are shown in algorithm 7 with the double boxed codes. Under the DDH assumption, we can get that there exists an efficient adversary B_5 :

$$\Pr[\text{G}_6 = 1] - \Pr[\text{G}_5 = 1] \leq \text{Adv}_{\mathsf{G},\mathsf{B5}}^{\text{DDH}}(\lambda)$$

Intuitively, in G_5 ,the value of $\text{XTemp}[w]$ is $g^{f_X(w)}$,which is the form of g^a if we replace $f_X(w)$ with a . The element in H is $\text{XTemp}[w]^{eid}$,that is $g^{f_X(w) \cdot eid}$,it is the form of g^{ab} if we replace eid

with b. With the above replacement, the distribution of H is indistinguishable from a random element in $\mathbb{G}$ under the DDH assumption. In the same way, Y is indistinguishable from a random element in $\mathbb{G}$.

Game G_7 **.** G_7 is almost same as G_6 except that it changes way in generating X, the details are shown in algorithm 8. In G_7, only elements in H that are used or accessed for multiple are added to X, otherwise, a random element in $\mathbb{G}$ is added to X. Furthermore, after H is generated,

Algorithm 7 G4, $\boxed{\text{G5}}$, $\boxed{\boxed{\text{G6}}}$

function INITIALIZE (DB, ACL, s, x, c, U)
$p, q, msk, mpk, g \xleftarrow{\$} \mathbb{Z}_n^*$
$f_I, f_X, f_Z \xleftarrow{\$} \mathrm{Fun}(\{0,1\}^\lambda, \mathbb{Z}_n^*)$
$\mathrm{T} \leftarrow \{\}; \mathrm{X} \leftarrow \{\}; cnt \leftarrow 0$; $(\mathrm{id}_i, \mathrm{W}_i) \leftarrow \mathrm{DB}$
for $w \in W$ and each id_i do
 $\mathrm{XTemp}[w] \leftarrow g^{f_X(w)}$; $\mathrm{eid} \leftarrow f_I(\mathrm{id}_i)$
 $\mathrm{H}[\mathrm{id}_i, w] \leftarrow \mathrm{XTemp}[w]^{\mathrm{eid}}$; $\boxed{\boxed{\mathrm{H}[\mathrm{id}_i, w] \xleftarrow{\$} \mathbb{G}}}$
for $w \in W$ do
 $(\mathrm{id}_1, ..., \mathrm{id}_{\mathrm{T}}) \leftarrow \mathrm{DB}[w]$; $cnt \leftarrow 1$
 $\mathrm{stag}_{\mathrm{w}} \xleftarrow{\$} \{0,1\}^\lambda$; $\mathrm{stags[w]} \leftarrow \mathrm{stag}_{\mathrm{w}}$
 $\sigma \xleftarrow{\$} \mathrm{Perm}([T_w])$; $\mathrm{WPerms[w]} \leftarrow \sigma$
 for $cnt \in [T_w]$ do
  $\mathsf{l} \leftarrow H(\mathrm{stag}_{\mathrm{w}} \| cnt)$; $u \leftarrow \mathrm{ABE.Enc}(mpk, 0^\lambda, \mathsf{U})$
  $eid \leftarrow f_I(id_{\sigma[cnt]})$; $z \leftarrow f_z(w \| cnt)$
  $v \leftarrow eid \cdot z^{-1}$; $\boxed{v \xleftarrow{\$} \mathbb{Z}_p^*}$
  $o \leftarrow op \oplus \mathsf{l}$; $\mathrm{T}[\mathsf{l}] \leftarrow (u, v, o)$
 end for
for $y \in \mathrm{W} \setminus w$ do
 for $cnt = T_w + 1, ..., T$ do
  $Y[w, y, cnt] \leftarrow X[y]^{f_Z(w\|cnt)}$
  $\boxed{\boxed{Y[w, y, cnt] \xleftarrow{\$} \mathbb{G}}}$
 end for
end for
 $\mathrm{X} \leftarrow \mathrm{XSETSETUP(DB,H)}$
 $\mathrm{EDB} \leftarrow \{\mathrm{T,X}\}$
for $k \in [T]$ do
 $\mathrm{t}[k] \leftarrow (\mathrm{DB, EDB, A}, H, \mathrm{s}[k], \mathrm{x}[k, \cdot])$
 end for
end function

function XSETSETUP(DB,H)
 $(\mathrm{id}_i, \mathrm{W}_i) \leftarrow \mathrm{DB}$; $\mathrm{X} \leftarrow \phi$
 for $w \in \mathrm{W}$ and $\mathrm{ind} \in \mathrm{DB}(w)$ do
 $x \leftarrow \mathrm{H}[id, w]$; $x \leftarrow g^{f_X(w) \cdot eid}$
 $\mathrm{X} \leftarrow \mathrm{X} \cup x$
 end for
end function

function TRANSGEN ($\mathrm{DB, EDB, A}, H, \mathrm{s}[k], \mathrm{x}[k, \cdot]$ **)**
$\alpha_i \leftarrow \mathrm{A}[i]$; $\mathrm{acf} \leftarrow \mathrm{stags}[\mathrm{s}_k]$
$(id_1, ..., \mathrm{id}_{\mathrm{T}}) \leftarrow \mathrm{DB}(s_k)$; $\sigma \leftarrow \mathrm{WPerms}[\mathrm{s_k}]$
for $\beta \in [|\mathrm{x}_k|]$ do
 for $cnt \in [T_s]$ do
  $l \leftarrow F(acf, cnt)$; $(u, v, o) \leftarrow \mathrm{EDB}[l]$
  $\mathrm{xtoken}[\beta, cnt] \leftarrow H[id_{\sigma[cnt]}, \mathrm{x[k}, \beta]]^{1/v}$
 end for
 for $cnt = T_s + 1, ..., T_c$ do
  $\mathrm{xtoken}[\beta, cnt] \leftarrow Y[\mathrm{s}_t, \mathrm{x[k}, \beta], cnt]$
 end for
 $\mathrm{ctl}[\beta] \leftarrow \alpha_i \cdot \mathrm{x[k}, \beta]$
end for
(stag,Res) $\leftarrow$ SEARCH(EDB,X,acf,xtoken,ctl)
ResInds $\leftarrow \mathrm{DB(s}[k]) \cap \mathrm{DB(x}[k,1]) \cap \mathrm{DB(c}[k,1])$
$... \cap \mathrm{DB(x}[k,\mathrm{n}]) \cap \mathrm{DB(c}[k,\mathrm{n}])$
return (stag,xtoken,Res,ResInds)
end function

XSETUP will just access H once, only the function TransGen access elements in H. However, elements that are accessed by TransGen satisfy that $\mathrm{id} \in \mathrm{DB}[\mathrm{s}_k]$ and $w = x[k, \beta]$. As for others, it is indistinguishable with random selection. Therefore, the distribution of G_7 is the same as G_6 ,so we get:

$$\Pr[G_7 = 1] = \Pr[G_6 = 1]$$

Game G_8 **.** G_8 is almost same as G_7 except the way to access H in function TransGen , as shown in algorithm 8. To test a possible repeated access to elements in H, the check is necessary that if either XSETUP will access the index or the function TransGen will read it again. In this case, XSETUP only access an index if $\mathrm{id} \in \mathrm{DB}(\mathrm{s}_k)$ and $\mathrm{x}[k, \beta] = w$ in G_7 , which meets the purpose of the first "if" in G_8. However, it is also possible in TransGen when there are two different queries, k and k'. For this situation, it should ensure $\mathrm{id} \in \mathrm{DB}[\mathrm{s}_k] \cap \mathrm{DB}[\mathrm{s}_{k'}]$ and $w = \mathrm{x}[k,\beta] \in \mathrm{x}_{k'}$, that is what the "else if" statement in G_8. Obviously,

$$\Pr[G_8 = 1] = \Pr[G_7 = 1].$$

Simulator: Simulator S takes $\mathsf{L}(\mathrm{DB},\mathrm{s},\mathrm{x}) = (N, \bar{s}, \mathrm{SN}, \mathrm{AN}, \mathrm{IP}, \mathrm{XN})$ as input and outputs a simulated EDB=(T,X) and a transcript array t. We prove that the simulator S and G_8 are indistinguishable, so we can prove that the simulator S satisfies theorem 1 through the transitivity of trust between games.

Algorithm 8 G_7, G_8

function INITIALIZE (DB, ACL, s, x, c, $\cup$) // G_7, G_8
$p, q, msk, mpk, g \xleftarrow{\$} \mathsf{Z}_n^*$
$f_I, f_X, f_Z \xleftarrow{\$} \mathrm{Fun}(\{0,1\}^\lambda, \mathsf{Z}_p^*)$
$\mathrm{T} \leftarrow \{\}; \mathrm{X} \leftarrow \{\}; cnt \leftarrow 0$; $(\mathrm{id}_i, \mathrm{W}_i) \leftarrow \mathrm{DB}$
for $w \in W$ and each id_i do
  $\mathrm{H}[\mathrm{id}_i, w] \xleftarrow{\$} \mathsf{G}$
end for
for $w \in \mathrm{s}$ do
  $\sigma \xleftarrow{\$} \mathrm{Perm}([T_s])$; $\mathrm{WPerms}[\mathrm{w}] \leftarrow \sigma$
end for
for $w \in W$ do
  $(\mathrm{id}_1, ..., \mathrm{id}_{\mathrm{T}_w}) \leftarrow \mathrm{DB}[w]$; $cnt \leftarrow 1$
  $\mathrm{stag}_{\mathrm{w}} \xleftarrow{\$} \{0,1\}^\lambda$; $\mathrm{stags}[\mathrm{w}] \leftarrow \mathrm{stag}_{\mathrm{w}}$
  for $cnt \in [T_w]$ do
    $\mathsf{l} \leftarrow H(\mathrm{stag}_{\mathrm{w}} \| cnt)$; $u \leftarrow \mathrm{ABE.Enc}(mpk, 0^\lambda, \cup)$
    $eid \leftarrow f_I[id_{\sigma[cnt]}]$; $z \leftarrow f_Z[w \| cnt]$
    $v \xleftarrow{\$} \mathsf{Z}_n^*$; $o \leftarrow op \oplus \mathsf{l}$
    $\mathrm{T}[\mathsf{l}] \leftarrow (\mathrm{u}, v, o)$
  end for
  for $y \in \mathrm{W} \setminus w$ do
    for $cnt = T_w + 1, ..., T$ do
      $Y[w, y, cnt] \leftarrow X[y]^{f_Z(w\|cnt)}$; $Y[w, y, cnt] \xleftarrow{\$} \mathsf{G}$
    end for
  end for
end for
$\mathrm{X} \leftarrow \mathrm{XSETSETUP(DB,H)}$
$\mathrm{EDB} \leftarrow \{\mathrm{T},\mathrm{X}\}$
for $k \in [T]$ do
  $\mathrm{t}[k] \leftarrow (\mathrm{DB}, \mathrm{EDB}, \mathrm{A}, H, \mathrm{s}[k], \mathrm{x}[k,\cdot])$
end for
end function

function XSETSETUP(DB,H) // **G_7, G_8**
  $(\mathrm{id}_i, \mathrm{W}_i) \leftarrow \mathrm{DB}$; $\mathrm{X} \leftarrow \phi$
  for $w \in \mathrm{W}$ and $\mathrm{ind} \in \mathrm{DB}(w)$ do
    if $\exists k : \mathrm{id} \in \mathrm{DB}(\mathrm{s}[k]) \wedge \mathrm{x}[k,\beta] = w$ then
      $x \leftarrow \mathrm{H}[id, w]$; $\mathrm{X} \leftarrow \mathrm{X} \cup x$
    else
      $h \xleftarrow{\$} \mathsf{G}$; $\mathrm{X} \leftarrow \mathrm{X} \cup \{h\}$
  end for
return X
end function

function TRANSGEN ($\mathrm{DB}, \mathrm{EDB}, \mathrm{A}, H, \mathrm{s}[k], \mathrm{x}[k,\cdot]$ **)**
//only **G_8**
$\alpha_i \leftarrow \mathrm{A}[i]$; $\mathrm{acf} \leftarrow \mathrm{stags}[\mathrm{s}_k]$
$(id_1, ..., \mathrm{id}_{\mathrm{T}_s}) \leftarrow \mathrm{DB}(s_k)$; $\sigma \leftarrow \mathrm{WPerms}[\mathrm{s}_k]$
for $\beta \in [|\mathrm{x}_k|]$ do
  for $cnt \in [T_s]$ do
  $l \leftarrow F(acf, cnt)$; $(u, v, o) \leftarrow \mathrm{EDB}[l]$
    if $\mathrm{id}_{\sigma[cnt]} \in \mathrm{DB}[\mathrm{s}_t] \cap \mathrm{DB}[\mathrm{x}[\mathrm{k},\beta]]$ then
  $\mathrm{xtoken}[\beta, cnt] \leftarrow H[id_{\sigma[cnt]}, \mathrm{x}[k,\beta]]^{1/v}$
    else if $\exists k' \neq k : \mathrm{id}_{\sigma[\mathrm{cnt}]} \in \mathrm{DB}[\mathrm{s}_{k'}] \wedge \mathrm{x}[k,\beta] \in \mathrm{x}_{k'}$ then
      $\mathrm{xtoken}[\beta, cnt] \leftarrow H[id_{\sigma[cnt]}, \mathrm{x}[k,\beta]]^{1/v}$
    else $\mathrm{xtoken}[\beta, cnt] \xleftarrow{\$} \mathsf{G}$
  end for
  for $cnt = T_s + 1, ..., T_c$ do
    $\mathrm{xtoken}[\beta, cnt] \xleftarrow{\$} \mathsf{G}$
  end for
  $\mathrm{ctl}[\beta] \leftarrow \alpha_i \cdot \mathrm{x}[k,\beta]$
end for
$(\mathrm{stag},\mathrm{Res}) \leftarrow \mathrm{SEARCH(EDB,X,acf,xtoken,ctl)}$
$\mathrm{ResInds} \leftarrow \mathrm{DB}(s[k]) \cap \mathrm{DB}(x[k,1]) \cap \mathrm{DB}(c[k,1])$
  $... \cap \mathrm{DB}(x[k,n]) \cap \mathrm{DB}(c[k,n])$
return (stag,xtoken,Res,ResInds)
end function

Firstly, S computes $\bar{\mathrm{x}}$, which is a restricted equality pattern of x, it denotes the server knows which xtrems are equal. With the elements in X, it is possible for the server to infer some certain xterms are equal because if there is a id that satisfies $\mathrm{id} \in \mathrm{DB}[\mathrm{s}[k_1]] \cap \mathrm{DB}[\mathrm{s}[k_2]]$, in which k_1 and k_2 are two different queries, then the server can infer $\mathrm{x}[k_1,\beta]$ is equal to $\mathrm{x}[k_2,\delta]$ due to the repeating values of elements in X that $\mathrm{xtemp}[\mathrm{x}[k_1,\beta],\mathrm{id}]$ and $\mathrm{xtemp}[\mathrm{x}[k_2,\delta],\mathrm{id}]$. This can be formulated equivalently in terms of the leakage IP by $\bar{\mathrm{x}}[k,\beta]$ such that $\bar{\mathrm{x}}[k_1,\beta] = \bar{\mathrm{x}}[k_2,\delta]$

iff $\mathrm{IP}[k_1,k_2] \neq \phi$. Particularly, we have: $\bar{\mathrm{x}}[k_1,\beta] = \bar{\mathrm{x}}[k_2,\delta] \Rightarrow \mathrm{x}[t_1,\beta] = \mathrm{x}[t_2,\delta]$ and $(\mathrm{x}[k_1,\beta] = \mathrm{x}[k_2,\delta]) \wedge (\mathrm{DB}[\mathrm{s}[k_1]] \cap \mathrm{DB}[\mathrm{s}[k_2]] \neq \phi) \Rightarrow \bar{\mathrm{x}}[k_1,\beta] = \bar{\mathrm{x}}[k_2,\delta]$.

To show that the distribution of S is same as G_8, we prove the distribution of EDB, X and xtoken is the same as that of G_8 ,respectively, the details of EDB in S are shown in algorithm 9. In S ,the generation of EDB is same as for G_8, in which $w \in \bar{\mathrm{s}}$ and $|\bar{\mathrm{s}}| < N$ which is obvious by the definition of $\bar{\mathrm{s}}$, so S fills out the additional random elements of EDB. In both G_8 and S, the elements in EDB are computed in the same way, so the distribution of S and G_8 is indistinguishable.

The X in simulator S is generated by algorithm 10. In both simulator and G_8 the elements in X are randomly chosen from group G . For the $\Sigma_{w \in W} \mathrm{DB}[\mathrm{w}]$, there are N elements, in G_8, the elements are added to X for $w \in \mathrm{W}$ and $\mathrm{ind} \in \mathrm{DB}(w)$. In S , this is done by keeping track of each

Algorithm 9 Generation of EDB in Simulator

$\mathrm{T} \leftarrow \{\}; cnt \leftarrow 0$
for $w \in \bar{\mathrm{s}}$ do
$\sigma \xleftarrow{\$} \mathrm{Perm}([\mathrm{SP}[k_1]])$; $\mathrm{WPerms}[\mathrm{w}] \leftarrow \sigma$
$\mathrm{stag_w} \xleftarrow{\$} \{0,1\}^\lambda$; $\mathrm{stags}[\mathrm{w}] \leftarrow \mathrm{stag_w}$
for $cnt \in [\mathrm{SP}[k_1]]$ do
$\mathsf{l} \leftarrow H(\mathrm{stag_w} \| cnt)$; $u \leftarrow \mathrm{ABE.Enc}(mpk, 0^\lambda, \mathsf{U})$
$v \xleftarrow{\$} \mathsf{Z}_p^*$; $o \leftarrow op \oplus \mathsf{l}$
$\mathrm{T}[\mathsf{l}] \leftarrow (u,v,o)$
end for
end for
for $k_2 = cnt+1,....,N$ do
$l \xleftarrow{\$} \{0,1\}^\lambda$; $u \leftarrow \mathrm{ABE.Enc}(mpk, 0^\lambda, \mathsf{U})$
$v \xleftarrow{\$} \mathsf{Z}_p^*$; $o \leftarrow op \oplus \mathsf{l}$
$\mathrm{T}[\mathsf{l}] \leftarrow (u,v,o)$
end for

Algorithm 10 Generation of X in Simulator

$\mathrm{X} \leftarrow \phi$; $k_2 \leftarrow 0$
for $w \in \bar{\mathrm{x}}$ and $\mathrm{id} \in \mathrm{U}_{k \in [\mathrm{T}], \beta \in [\mathrm{A}]} \mathrm{AN}[k,\beta]$ do
$\mathrm{H}[id, w] \xleftarrow{\$} \mathsf{G}$
for $w \in \bar{\mathrm{x}}$ and $\mathrm{id} \in \mathrm{U}_{\{(\mathrm{k},\beta):\bar{\mathrm{x}}[\mathrm{k},\beta]=\mathrm{w}\}} \mathrm{AN}[k,\beta]$ do
$x \leftarrow \mathrm{H}[id, w]$; $\mathrm{X} \leftarrow \mathrm{X} \cup x$
$k_2 \leftarrow k_2 + 1$
end for
for $k_1 = k_2+1,...,N$ do
$v \xleftarrow{\$} \mathsf{G}$; $\mathrm{X} \leftarrow \mathrm{X} \cup v$
end for

Algorithm 11 Generation of t in Simulator

for $\tau \in [T]$ do
for $w_x \in [\mathrm{XT}[\tau]]$ do
$R \leftarrow \mathrm{AN}[\tau, w_x] \cup \mathrm{U}_{\mathrm{k}' \in [\mathrm{T}], \delta \in [\mathrm{XT}[\tau]]} \mathrm{IP}[\tau, k', w_x, \delta]$
$cnt \leftarrow 1$
for $\mathrm{id} \in \mathrm{WPrems}[\bar{\mathrm{s}}[\tau]]$ do
$(\mathrm{id}_1, \mathrm{id}_2, ..., \mathrm{id}_{T_s}) \leftarrow \mathrm{DBT}[\tau]$
$\sigma \leftarrow \mathrm{WPerms}[\bar{\mathrm{s}}[\tau]]$
for $cnt \in [T_s]$ do
if $\mathrm{id}_{\sigma[cnt]} \in R$ then
$l \leftarrow H(\mathrm{stags}[\bar{\mathrm{s}}[\tau]], cnt)$
$(u,v,o) \leftarrow \mathrm{EDB}[l]$
$\mathrm{xtoken}[\tau, w_x, cnt] \leftarrow \mathrm{H}[\mathrm{id}_{\sigma[cnt]}, \bar{\mathrm{x}}[\tau, w_x]]^{1/\mathrm{v}}$
else
$\mathrm{xtoken}[\tau, w_x, cnt] \xleftarrow{\$} \mathsf{G}$
end if
$\mathrm{ctl}[\beta] \leftarrow \alpha_i \cdot \bar{\mathrm{x}}[\tau, \mathrm{w_x}]$; $cnt \leftarrow cnt + 1$
end for
end for
for $cnt = \mathrm{SN}[\bar{\mathrm{s}}[\tau]]+1,...,T_{cnt}$ do
$\mathrm{xtoken}[\tau, w_x, cnt] \xleftarrow{\$} \mathsf{G}$
end for
end for
(stag,Res) $\leftarrow$ SEARCH(EDB,X,acf,xtoken,ctl)
ResInds $\leftarrow$ (τ, RP,DBT)
return (stag,xtoken,Res,ResInds)
end for

addition with k_2 , and adding additional $(N-k_2)$ elements at last. For the distribution the X, we show that with the xtoken.

The transcript t including the xtokens in simulator S are generated by algorithm 11. The y and σ are being uniformly random , hence distributed identically both in G_8 and S . The reuse of σ is almost same in these two games, σ are reused when an sterm is repeated, while in S σ are reused when $\bar{\mathrm{s}}$ repeats.

Next, we observe that in calculating xtokens in G_8, the H is accessed either the id satisfies a conjunction query that $\mathrm{id} \in \mathrm{DB}[\mathrm{s}_k] \cap \mathrm{DB}[\mathrm{x}[k,\beta]]$ or the id is in another query with the same xterm. The simulator S has the same logic by reading the $\sigma(cnt)$-th identifier in R which contains these two conditions.

Finally, we show that the reusage of H is same in G_8 and S when H is used for multiple times. Consider two elements of H that $(\mathrm{id}_1, \mathrm{x}[k_1,\beta])$ and $(\mathrm{id}_2, \mathrm{x}[k_2,\beta])$ are read in different queries in G_8, so id_1 either satisfies the conjunction query or another query with the same xtrem, so as to id_2. In S ,the simulator will read values from RP or IP where the indices are same with $(\mathrm{id}_1, \overline{\mathrm{x}}[k_1,\beta])$and $(\mathrm{id}_2, \overline{\mathrm{x}}[k_2,\beta])$ in H. We claim that:

$$(\mathrm{id}_1, \mathrm{x}[k_1,\beta]) = (\mathrm{id}_2, \mathrm{x}[k_2,\beta]) \Leftrightarrow (\mathrm{id}_1, \overline{\mathrm{x}}[k_1,\beta]) = (\mathrm{id}_2, \overline{\mathrm{x}}[k_2,\beta]) \qquad (1)$$

The left direction $\Leftarrow$ of (1) is easy since that for $\overline{\mathrm{x}}$,we have

$$\overline{\mathrm{x}}[k_1,\beta] = \overline{\mathrm{x}}[k_2,\delta] \Rightarrow \mathrm{x}[k_1,\beta] = \mathrm{x}[k_2,\delta],$$

and $\overline{\mathrm{x}}$ has another property that $(\mathrm{x}[k_1,\beta] = \mathrm{x}[k_2,\beta]) \wedge (\mathrm{DB}[\mathrm{s}[k_1]] \cap \mathrm{DB}[\mathrm{s}[k_2]] \neq \phi) \Rightarrow \overline{\mathrm{x}}[k_1,\beta] = \overline{\mathrm{x}}[k_2,\delta]$, if $\mathrm{DB}[\mathrm{s}[k_1]] \cap \mathrm{DB}[\mathrm{s}[k_2]] \neq \phi$,then the direction $\Rightarrow$ will be proven. Suppose that $(\mathrm{id}_1,\mathrm{x}[k_1,\beta])$ = $(\mathrm{id}_2,\mathrm{x}[k_2,\beta])$,we have $\mathrm{id}_1 = \mathrm{id}_2$,but this means the id is in $\mathrm{DB}[\mathrm{s}[k_1]] \cap \mathrm{DB}[\mathrm{s}[k_2]]$ and the intersection is not empty, thus we have $\overline{\mathrm{x}}[k_1,\beta] = \overline{\mathrm{x}}[k_2,\beta]$.

5. Performance Analysis

To evaluate the performance of our DMC-SSE scheme, we conduct experiments based on real data set, and compared it with multi-client schemes Du [23] and Jarecki [19] that with similar functions from two aspects of function and performance.

5.1 Functional Analysis

Table 1. Comparison of functionality features

	multi-client	**dynamic**	**file update**	**Boolean query**
Jarecki [19]	**Yes**	**No**	**No**	**Yes**
Du [23]	**Yes**	**Yes**	**No**	**Yes**
Our scheme	**Yes**	**Yes**	**Yes**	**Yes**

First of all, we conduct a functional comparative analysis as shown in **Table 1**, although all three schemes implement multi-client SSE and support Boolean queries, but Du [23] and our scheme can better update the clients dynamically. As for the dynamic update of the client, both ACL and index are needed to be updated in the scheme proposed by Du, unlike this, only ACL is updated in our scheme. At last, our scheme supports dynamic updates of files, which is not supported by the other two schemes.

Table 2. Comparison of computing cost

	EDBSetup	**TokenGen**	**Search**
Jarecki [19]	$O(N_p \cdot \mathrm{ep})$	$O(N_p \cdot N_c \cdot \mathrm{ep})$	$O(N_c \cdot \mathrm{ep})$
Du [23]	$O(N_p \cdot \mathrm{ep} + N_p \cdot N_f \cdot \mathrm{bp})$	$O(N_p \cdot \mathrm{ep})$	$O(N_c \cdot \mathrm{bp})$
Our scheme	$O(2N_p \cdot \mathrm{ep})$	$O(N_p \cdot N_c \cdot \mathrm{ep})$	$O(N_c \cdot \mathrm{ep})$

As for computing overhead, only the most time-consuming operations are considered: exponential operation (denoted by ep) and bilinear pairing (denoted by bp), therefore, only the calculation cost of EDBSetup, TokenGen and Search algorithms are compared, other algorithms that use fewer of these operations are omited, i.e., ClientAuth. The comparison results are shown in **Table 2**, in which N_c denotes the number of clients and N_f denotes the number of files corresponding to the keyword w ,besides, N_p is the total number of keyword-identifier pairs, $N_p = |\mathrm{DB(w)}|$. In our scheme, EDBSetup needs to computes 1 exponentiation to realize the attribute encryption of the symmetric key and the file identification, thus there are 2 ep in one keyword-identifier pair.

Table 3. Comparison of communication cost

	Token size	**Database size**
Jarecki [19]	$O(N_q \cdot N_c \cdot \vert \mathbb{G}_1 \vert)$	$O(N_p \cdot (\lambda + \vert \mathbb{Z}_p^* \vert + \vert \mathbb{G}_1 \vert))$
Du [23]	$O(N_q \cdot \vert \mathbb{G}_1 \vert)$	$O(N_p \cdot (\lambda + \vert \mathbb{G}_2 \vert) + N_p \cdot N_f \cdot \vert \mathbb{G}_T \vert)$
Our scheme	$O(N_q \cdot N_c \cdot \vert \mathbb{G}_1 \vert)$	$O(N_p \cdot (\lambda + \vert \mathbb{Z}_p^* \vert + \vert \mathbb{G}_1 \vert))$

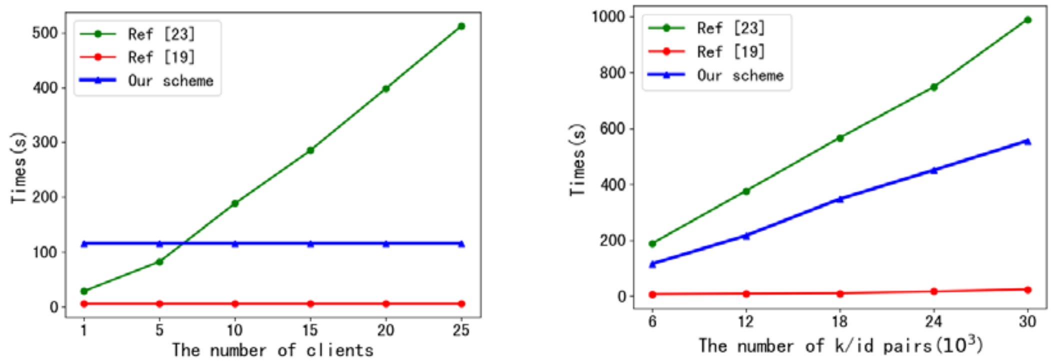
Since the communication cost mainly depends on the size of the data transmitted in the network, the size of the data can be used to evaluate the communication cost. In our scheme, the data transmitted in the network is mainly the encrypted database and the search token generated during keyword query, so we use the size of them to evaluate the communication cos like Du [23], the comparison result is shown in **Table 3**. In which $|\cdot|$ denotes the size of a set or group and N_q is the number of keywords to query.

5.2 Performance Analysis

We deploy our experiments on a local machine with an operating system of Ubuntu 18.04, Intel(R) Core (TM) i7-8550U CPU and 8 GB of RAM. We use python 3.6 to compile our programs on Pycharm 2020.2. We use charm-crypto library to implement cryptographic and group operations. For PRFs and hash functions, we use AES-128 and HMAC-MD5 respectively, also NIST 224p elliptic curves is used for group operations.

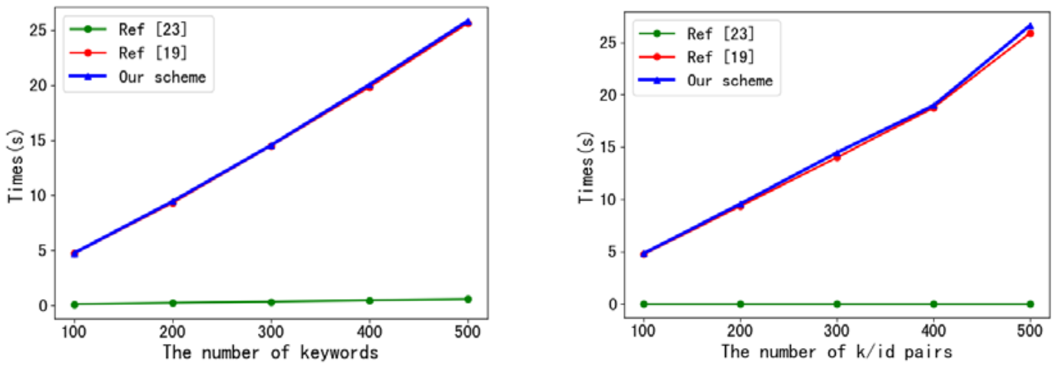
For dataset, we adopt Enron Email Dataset in our experiments. Enron Email Dataset has about 517401 email files from about 150 users. Keywords are extracted by the jieba library in python, and about 1672878 keywords are extracted and we generate an inverted index based on the extracted keywords, our experiments are mainly based on this index.

We analyze the experimental results of the three algorithms of EDBSetup, TokenGen, and Search, which are computationally expensive, to evaluate the performance of our scheme.



(a) EDBSetup **(b) EDBSetup**

Fig. 3. The time cost in EDBSetup. (a) the number of keyword-identifier is fixed at 6000 with various clients. (b) the number of clients is fixed at 10 with various keyword-identifier.

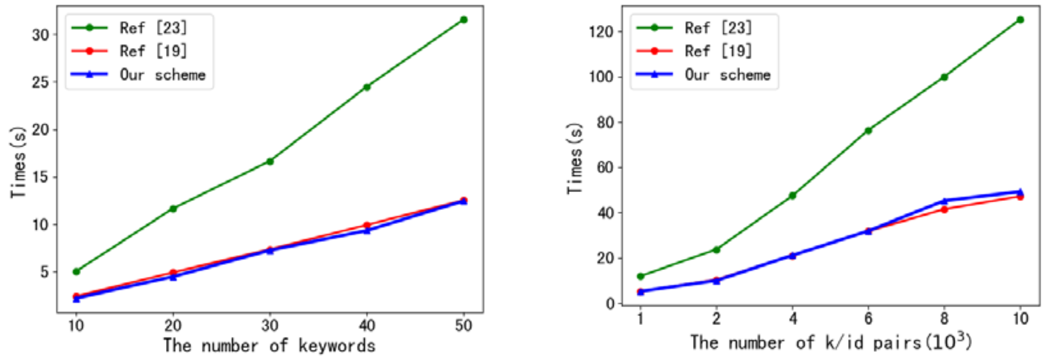


(a) TokenGen **(b) TokenGen**

Fig. 4. The time cost in TokenGen. (a) the number of keyword-identifier is fixed at 200 with various keywords. (b) the number of keywords is fixed at 100 with various keyword-identifier.

Fig. 3 shows the comparison results of the time cost in EDBSetup. In **Fig. 3** (a), the scheme proposed by Du grows linearly with the number of clients, however, it has almost no impact on our scheme and the scheme proposed by Jarecki [19]. The reason is that Du [23] performs bilinear map operation for each client, which is not needed in our scheme. In **Fig. 3** (b), all three schemes grow linearly with the number of keyword-identifier pairs, but our scheme takes less time than Jarecki [23].

Fig. 4 shows the comparison results of the time cost in TokenGen. Our scheme and Jarecki [19] cost more time than Du [23] since the use bilinear mapping makes the number of tokens only depends on the number of keywords in Du [23]. The time cost of our scheme is almost the same as that of Jarecki [19], but it is more because our scheme needs to calculate the blind factor $\mathrm{ctl}[j] \leftarrow \alpha_i \cdot w_j$ when generating the token.



(a) Search **(b) Search**

Fig. 5. The time cost in Search. (a) the number of keyword-identifier is fixed at 1000 with various keywords. (b) the number of keywords is fixed at 20 with various keyword-identifier.

Fig. 5 shows the time cost spent on a Boolean query in the three schemes, which grows linearly with the number of keywords and files in the above three schemes. The scheme of Du [23] has the highest time cost due to expensive bilinear pairing operation, since our scheme performs file filtering operations, our solution takes more time than Jarecki [19].

6. Conclusion

In this paper, we propose a searchable symmetric encryption scheme for multi-client that supports Boolean queries, DMC-SSE, which realizes multi-keyword search in multi-client scenario and supports fine-grained access control of client, in addition, our scheme realizes full dynamic update of client and file. Experimental results and security analysis show that our scheme is correct, efficient and secure.

References

[1] D. X. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in *Proc. of IEEE Symp. Secur. Privacy*, pp. 44–55, 2000. Article(CrossRef Link)

[2] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in *Proc. of Int. Conf. Theory Appl. Cryptographic Techn.*, pp. 506–522, 2004. Article(CrossRef Link)

[3] M.Naveed, M. Prabhakaran, and C.A. Gunter, “Dynamic searchable encryption via blind storage,” in *Proc. of IEEE Symp. Secur. Privacy*, pp. 639-654, 2014. Article(CrossRef Link)

[4] S. Kamara, C. Papamanthou, and T. Roeder, “Dynamic searchable symmetric encryption,” in *Proc. of ACM Conf. Comput. Commun. Secur.*, pp. 965–976, 2012. Article(CrossRef Link)

[5] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W. Kim, “Forward secure dynamic searchable symmetric encryption with efficient updates,” in *Proc. of ACM Conf. Comput. Commun. Secur.*, pp. 1449–1463, 2017. Article(CrossRef Link)

[6] J. Li, Y. Huang, Y. Wei, Z. L. Liu, C. Y. Dong, W. J. Lou, “Searchable Symmetric Encryption with Forward Search Privacy,” *IEEE Trans. Dependa. Secure Comput*, vol. 18, no. 1, pp. 460-474, Jan/Feb 2021. Article(CrossRef Link)

[7] S. Tahir, S. Ruj, Y. Rahulamathavan, M. Rajarajan and C. Glackin, "A New Secure and Lightweight Searchable Encryption Scheme over Encrypted Cloud Data," *IEEE Trans. Emerging Topics in Computing*, vol. 7, no. 4, pp. 530-544, 1 Oct.-Dec. 2019. Article(CrossRef Link)

[8] H. Li, Y. Yang, Y. Dai, S. Yu and Y. Xiang, "Achieving Secure and Efficient Dynamic Searchable Symmetric Encryption over Medical Cloud Data," *IEEE Trans.Cloud Comput*, vol. 8, no. 2, pp. 484-494, 1 April-June 2020. Article(CrossRef Link)
[9] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895-934, 2011. Article(CrossRef Link)
[10] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. of Advances in Cryptology - ASIACRYPT 2010*, pp. 577–594, 2010. Article(CrossRef Link)
[11] S. Kamara, T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Proc. of Advances in Cryptology – EUROCRYPT 2017*, pp.94-124, 2017. Article(CrossRef Link)
[12] B. Fuhry, R. Bahmani, F. Brasser, F. Hahn, F. Kerschbaum, AR. Sadeghi, "HardIDX: Practical and Secure Index with SGX," in *Proc. of IFIP Annual Conference on Data and Applications Security and Privacy XXXI*, pp.386-408, 2017. Article(CrossRef Link)
[13] P. Golle, J. Staddon, and B. R. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. of International Conference on Applied Cryptography and Network Security*, pp. 31–45, 2004. Article(CrossRef Link)
[14] L. Ballard, S. Kamara, F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. of 7th. Int. Conf. Info. Commu. Secur*, pp. 414–426, 2005. Article(CrossRef Link)
[15] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," *Advances in Cryptology-CRYPTO*, pp. 353–373, 2013. Article(CrossRef Link)
[16] S. Lai, S. Patranabis, A. Sakzad, J. Liu, D. Mukhopadhyay, R. Steinfeld, S. Sun, D. Liu, and C. Zuo, "Result pattern hidingsearchable encryption for conjunctive queries," in *Proc. of ACM Conf. Comput. Commun. Secur.*, pp. 745–762, 2018. Article(CrossRef Link)
[17] G. Xu, H.W. Li, Y.S. Dai, K. Yang, X.D. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans.Inf. Forensics Security*, vol.14, no.4, pp.870-885, Apr.2019. Article(CrossRef Link)
[18] M. Raykova, B. Vo, S. M. Bellovin, and T. Malkin, "Secure anonymous database search," in *Proc. of the 2009 ACM workshop on Cloud computing security*, pp. 115–126, 2009. Article(CrossRef Link)
[19] S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proc. of ACM Conf. Comput. Commun. Secur.*, pp. 875–888, 2013. Article(CrossRef Link)
[20] B.A. Fisch, B. Vo, F. Krell, A. Kumarasubramanian, V. Kolesnikov, T. Malkin, S.M. Bellovin, "Malicious-client security in blind seer: a scalable private DBMS," in *Proc. of IEEE Symp. Secur. Privacy*, pp. 395–410, 2015. Article(CrossRef Link)
[21] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, M. Steiner, "Rich queries on encrypted data: beyond exact matches," in *Proc. of ESORICS*, Vienna, Austria, 123–145, 2015. Article(CrossRef Link)
[22] S.F. Sun, C. Zuo, J.K. Liu, A. Sakzad, R. Steinfeld, T.H. Yuen, D. Gu, "Non-Interactive Multi-Client Searchable Encryption: Realization and Implementation," *IEEE Trans. Depend. Secure Comput*, vol. 19, no. 1, pp. 452-467, 2022. Article(CrossRef Link)
[23] L. Du, K. Li, Q. Liu, Z. Wu, S. Zhang, "Dynamic multi-client searchable symmetric encryption with support for boolean queries," *Inf. Sci*, vol.506, pp.234-257, Jan. 2020. Article(CrossRef Link)
[24] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. of 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, pp. 668-679, 2015. Article(CrossRef Link)
[25] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. of IEEE Symp. Secur. Privacy*, pp. 707-720, 2016. Article(CrossRef Link)

[26] Y. Wei, S. Lv, X. Guo, Z. Liu, Y. Huang, and B. Li, "FSSE: Forward secure searchable encryption with keyed-block chains," *Inf. Sci.*, vol. 500, pp. 113-126, Oct. 2019. Article(CrossRef Link)
[27] J. G. Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "New constructions for forward and backward private symmetric searchable encryption," in *Proc. of ACM Conf Computer Commun Secur.*, pp. 1038-1055, 2018. Article(CrossRef Link)
[28] X. Song, C. Dong, D. Yuan, Q.L. Xu, M.H. Zhao, "Forward Private Searchable Symmetric Encryption with Optimized I/O Efficiency," *IEEE Trans. Depend. Secure Comput*, vol.17, no.5, pp.912-927, Sept.-Oct. 1 2020. Article(CrossRef Link)
[29] H. Li, Y. Yang, Y. Dai, Y. Shui, X. Yong, "Achieving Secure and Efficient Dynamic Searchable Symmetric Encryption over Medical Cloud Data," *IEEE Trans.Cloud Comput*, vol.8, no.2, pp. 484-494. April-June 2020. Article(CrossRef Link)
[30] X. Liu, G. Yang, Y. Mu, H. Deng, "Multi-user Verifiable Searchable Symmetric Encryption for Cloud Storage," *IEEE Trans. Depend.Secure Comput*, vol.17, no.6, pp.1322-1332, Nov.-Dec. 1 2020. Article(CrossRef Link)

Wanshan Xu is a Ph.D.candidate in Beijing University of Technology. He received his M.S.degree in computer technology from Beijing University of Technology, Beijing, China, in 2015. His research interests include information security, data security, blockchain.

Jianbiao Zhang (Member, IEEE) received the B.S., M.S. and Ph.D. degree in computer science, all from Northwestern Polytechnic University in 1992, 1995 and 1999, Xi'an, China. From 1999 to 2001, he was a postdoctoral fellow in Beijing University of Aeronautics and Astronautics, Beijing, China. Now, he is a professor and Ph.D. supervisor in Faculty of Information Technology, Beijing University of Technology. His research interests include network and information security, trusted computing. He has published over 80 journal/conference papers.

Yilin Yuan received the B.S. and M.S. degree in college of computer and information engineering from Henan Normal University, Xinxiang, China, in 2015 and 2018 respectively. She is currently pursuing the Ph.D. degree in college of computing at Beijing University of Technology, Beijing, China. Her research interests include cloud security and trusted computing