DMU-ABSE: Dynamic Multi-user Attribute-Based Searchable Encryption with File Deletion and User Revocation

Jiming Liu[1], Zhenfu Cao[1,2(✉)], Xiaolei Dong[1], and Jiachen Shen[1(✉)]

[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
51174500107@stu.ecnu.edu.cn, {zfcao,dongxiaolei,jcshen}@sei.ecnu.edu.cn
[2] Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai, China

Abstract. Searchable encryption (SE) is a new cryptographic technique that allows data users searching for the files of their interests over huge amounts of encrypted files on the cloud. When it comes to multi-user setting, more issues should be addressed comparing to single-user setting, including key distribution, search privilege control and access control. In this paper, we propose DMU-ABSE, a dynamic multi-user ciphertext-policy attribute-based searchable encryption scheme with file deletion and user revocation. We manipulate an attribute-based encryption to achieve fine-grained search privilege control and hidden policy in multi-user setting while searching time of the proposed scheme is constant ($O(1)$). With the help of proxy re-encryption, we build one searchable index matrix by different owners in order to improve the searching efficiency. Furthermore, our scheme implements access control by embedding decryption keys into the index matrix. The proposed scheme is proved IND-CKA and IND-CPA semantically secure and experimental results shows that our scheme is efficient.

Keywords: Dynamic multi-user searchable encryption · Cipher-policy attribute-based encryption · Proxy re-encryption

1 Introduction

With the rapid growth of cloud computing technology, an increasing number of files, including documents, emails, videos, music and so on, have been outsourced to third-party cloud servers to make full use of powerful calculation capacity and massive storage space of cloud. These outsourced data usually contains sensitive information and business secrets which should not be leaked to the dishonest cloud server. To ensure data confidentiality and user privacy, data owners have to encrypt their data before outsourcing them to cloud. However, it is usually difficult to perform computation or search tasks over encrypted data. It is also infeasible for a user to download the entire archive when he needs to

W. Meng and S. Furnell (Eds.): SocialSec 2019, CCIS 1095, pp. 17–31, 2019.
https://doi.org/10.1007/978-981-15-0758-8_2

search within it. Searchable encryption is proposed to solve this problem [15]. It enables users to search over encrypted data that have been outsourced to the cloud servers. In searchable encryption, data owners encrypt the files and generate associated indexes, which can be used to search over the whole data set without leaking information, before upload them to the cloud server [1]. When a data user wants to search for the files of his interest, he first generates a search trapdoor with keyword and then submits it to the cloud server. Finally, the cloud server performs the search operation and return the matched files to the data user without leaking any information [4]. According to the number of data owners and data users, SE schemes can be categorized into four models: one-to-one SE, one-to-many SE, many-to-one SE, and many-to-many SE.

When it comes to the many-to-many scenario, which is also called multi-user searchable encryption (MUSE) [6], access control is usually required to manage who is allowed to search and access the data. For example, different position in a company or university should be granted different search privileges over different files. Attribute-based encryption (ABE), in which data user is defined by a set of attributes, is known as a typical way to achieve fine-grained access control. There are two types of ABE schemes: key-policy ABE (KP-ABE) [7] and ciphertext-policy ABE (CP-ABE) [11], distinguished by the access policy embed in whether the ciphertext or the private key. In 2014, attribute-based encryption searchable encryption (ABSE) was introduced by Khader [9], and he discussed the security of the ABSE scheme. In ABSE schemes, only the data users whose attribute set satisfies the access policy have the access to search for the corresponding keywords. Zheng [18] proposed a verifiable attribute-based keyword search (VABKS), which enables user to verify whether the cloud has faithfully executed the search operation and return the true search result. In 2016, Sun [16] proposed a more general construction which supports user revocation and expressive search capability. However, the access policy is exposed in the air in [16]. In 2017, Wang [17] proposed a multi-value-independent ABKS scheme which makes the search time constant, irrelevant to the number of attributes, and hides the access policy. However it does not support dynamic update. Moreover, comparing to the single-user scheme, the multi-user scheme faces a series number of additional challenges. For example, how to distribute different private keys of different users and prevent collusion between users, and meanwhile keep high search efficiency in the same time is a challenging problem.

In this paper, we focus on the multi-user searchable encryption scheme which enables dynamic deletion operation while keeping highly efficiency and semantic security and propose a dynamic multi-user attribute-based searchable encryption, DMU-ABSE. In order to support access control, we improve an attribute-based encryption [17] to achieve fine-grained access control in multi-user searchable encryption, which is hidden-policy, constant size and non-deterministic. Besides, inspired by a single-user dynamic searchable encryption [8], we adopt index matrix and proxy re-encryption (PRE) to improve the proposed multi-user searchable scheme with respect to allowing update and deletion operations and reducing search time overhead. Proxy re-encryption (PRE) [3] is based on the

public-key system. In a PRE system, proxy server can use a proxy re-encryption key to transform the ciphertext to another ciphertext under different secret keys without leaking any information about the plaintext and keys [14]. At last we will give security proof of DMU-ABSE. Experiments will also be given to show that it is efficient. In summary, we contribute a multi-user attribute-based searchable encryption scheme that is:

1. Hidden policy: Existing ABE schemes usually store access policy in the plaintext form in the encrypted index. Our scheme hides the access policy to prevent cloud server from learning the exact access policy of each file.
2. Constant trapdoor: The length of search tokens in existing attribute-based searchable encryption is usually linear to the number of attributes. But they are irrelevant in the proposed scheme.
3. Non-deterministic: Search trapdoor is non-deterministic in our scheme so that the trapdoor generated for an arbitrary keyword varies every time.
4. User revocation & File deletion: Our scheme provides the revocation algorithm to revoke the search authority to a user and deletion algorithm of a file on the cloud to support dynamic archives.
5. Decryption function: In traditional searchable encryption schemes, the search function is separated from decryption function. Our scheme supports access control on both searching and decryption functions.
6. Security: Security proofs will be given to show that our scheme is IND-CKA and IND-CPA secure in the generic bilinear group model which prevents collusion between different users.

2 System Model

2.1 System Roles

Our scheme mainly consists of five entities, namely trusted authority, data owners, data users, cloud server and proxy server in Fig. 1. The trusted authority is responsible for generating independent secret keys for data owners and data users, and system public keys. Each data owner firstly uses symmetric encryption to encrypt his original files and generate corresponding keyword indexes embed with access policy. And then data owners send these encrypted files and keyword indexes to the proxy server. Then, the proxy server re-encrypts these indexes and sends to the cloud server. The cloud server stores the re-encrypted indexes into an index matrix. While a data user wants to search for the files with target keyword, he firstly creates a search trapdoor by the keyword and secret key, and then sends the trapdoor to the proxy server. The proxy server re-encrypts the trapdoor and sends it to the cloud server to search in the index matrix. Finally, the cloud server returns all the matched files and corresponding decryption keys to the data owner.

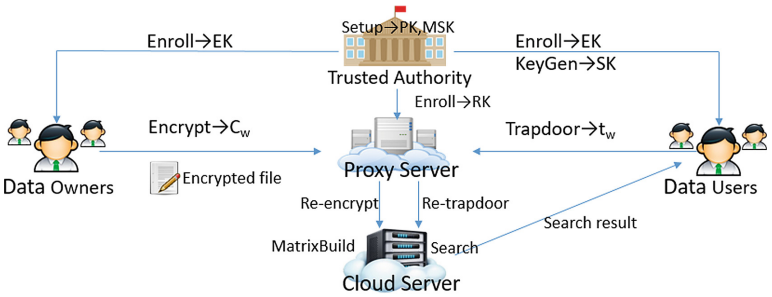


Fig. 1. System model

2.2 Application Scenario

The proposed scheme can be applied to many scenarios, such as personal health records (PHR) systems in which a huge amounts of patients updating their health records to the cloud and doctors can search for the target population or symptoms for research. Besides, it also can be applied to online subscription system where commercial publishers share charging data like videos or music with online subscribers.

2.3 Definition

- **Setup**$(1^\lambda) \rightarrow PK, MSK, MPK$
  Setup takes as input a security parameter λ, returns a public key PK, a master key MSK, and a master proxy key MPK.
- **Enroll**$(MSK, ID) \rightarrow EK_{ID}, RK_{ID}$
  Enroll algorithm is run by trusted authority, that takes as input the master key MSK and the new enrolled user ID. It returns enroll key EK_{ID} and corresponding proxy re-encryption key RK_{ID}.
- **KeyGenUser**$(MSK, S) \rightarrow SK$
  KeyGenUser algorithm is run by trusted authority, which takes as input the master key MSK and attributes set of a data user. It returns secret key SK for the data user.
- **Encrypt**$(PK, M \in \mathbb{G}_2, EK_{ID}, W, \mathbb{A}) \rightarrow C_w$
  Encrypt algorithm takes as input the public key PK, a decryption key of the file $M \in \mathbb{G}_1$, the enroll key EK_{ID}, keywords list $W = (w_1, w_2, ..., w_m)$ and the corresponding access structure $\mathbb{A}$, and generates an index ciphertext C_w. Only the secret keys satisfies $\mathrm{S} = \mathbb{A}$ have access to search and decrypt index ciphertext C_w. Besides, the original file is supposed to be encrypted in symmetric encryption such as AES with symmetric key $M \in \mathbb{G}_1$.
- **Re-Encrypt**$(MPK, C_w, RK_{ID}) \rightarrow C'_w$
  Re-Encrypt algorithm is run by proxy server, which takes as input the master proxy key MPK, an encrypted index C_w and the re-encryption key RK_{ID} of the user ID, and outputs the re-encrypted index C'_w which will be send to the cloud server.

- **MatrixBuild**$(C'_w, id, \mathbb{M}) \rightarrow \mathbb{M}'$
  MatrixBuild algorithm is run by cloud server after receiving the re-encrypted index C'_w and file id from proxy server. It inserts the encrypted indexes into the index matrix $\mathbb{M}$ in Fig. 2.
- **Trapdoor** $(SK, EK_{ID}, w) \rightarrow t_w$
  Trapdoor algorithm is run by data user, which takes as input the secret key SK and a keyword w, it returns a search token t_w.
- **Re-Trapdoor**$(MPK, t_w, RK_{ID}) \rightarrow t'_w$
  Re-encryption algorithm is run by proxy server, which takes as input the master proxy key MPK, a search token t_w, and returns a re-encrypted token t'_w and sends it to cloud server.
- **Search**$(t'_w, \mathbb{M}) \rightarrow \mathcal{R}$
  Search algorithm is run by the cloud server to search for the target files in the index matrix $\mathbb{M}$ with t'_w. It returns the list $\mathcal{R}$ of all matched files and corresponding index ciphertext.
- **Decrypt**$(SK, C_{id}) \rightarrow M$
  Decrypt algorithm takes as input an index ciphertext C_{id} and a secret key SK, and returns the message M if S satisfies $\mathbb{A}$ embed in the ciphertext C_{id}. S is the attribute set used to generate SK. And then M can used to decrypt the file id.
- **FileDeletion**$(id, \mathbb{M})$
  File-Deletion algorithm takes as input the id of the file to be deleted, and the cloud server deletes the file and all corresponding index blocks in the index matrix $\mathbb{M}$ logically.
- **Revocation**(ID)
  Because of graduation, retirement or anything else, when a user wants to leave the system, his search privilege should be revoked. Revocation algorithm takes as input the user ID, and revokes his update and search permission.

2.4 System Adversarial Model

We now define the security for DMU-ABSE in the sense of semantic security in the generic bilinear group model.

We assume that cloud server and proxy server are semi-honest-but-curious, while proxy server won't collude with cloud server or any user. Firstly, we need to ensure that re-encrypt index label L_p which is send to the cloud server does not reveal any information about w. We define security against an active attacker who is able to obtain trapdoors for any w of his choice. Even under such attack, the attacker should not be able to distinguish an encryption of a keyword w_0 from an encryption of a keyword w_1 for which he did not obtain the trapdoor. Formally, we define security against an active adversary $\mathcal{A}$ using the following game between a challenger $\mathcal{B}$ and the adversary $\mathcal{A}$:

1. The challenger $\mathcal{B}$ runs the **Setup** algorithm and give public parameters PK to adversary $\mathcal{A}$.

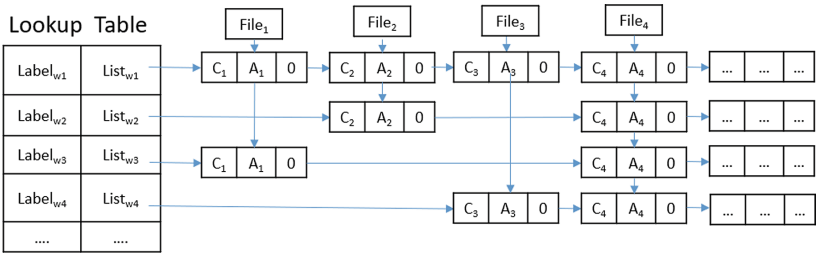


Fig. 2. Storage structure - Index matrix $\mathbb{M}$

2. The adversary $\mathcal{A}$ can adaptively ask the challenger $\mathcal{B}$ for the trapdoor for the keyword w
3. At some point, the adversary $\mathcal{A}$ sends the challenger $\mathcal{B}$ two keyword w_0, w_1 on which it wishes to be challenged. The challenger $\mathcal{B}$ picks a random $b \in \{0, 1\}$ and gives the adversary the challenge index label L_p^*.
4. Adversary $\mathcal{A}$ can continue to ask for re-trapdoor t'_w for any keyword w with the restriction that $w \neq w_0, w_1$.
5. Eventually, the adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and wins the game if $b = b'$

The advantage of an adversary is defined to be $Adv = |Pr[b = b'] - 1/2|$.

Definition 1. *(IND-CKA) DMU-ABSE is semantically secure against an adaptive chosen keyword attack if all probabilistic polynomial-time (PPT) attackers have at most negligible advantage in λ in the above security game.*

We now define security for DMU-ABSE to ensure that C_{id} does not reveal any information about M. We define security against an active attacker who is able to obtain secret keys which cannot decrypt the challenge ciphertext. Formally, we define security against an active adversary $\mathcal{A}$ using the following game between a challenger $\mathcal{B}$ and the adversary $\mathcal{A}$.

1. The challenger $\mathcal{B}$ runs the **Setup** algorithm and give public parameters PK to adversary $\mathcal{A}$.
2. The adversary $\mathcal{A}$ can adaptively ask the challenger $\mathcal{B}$ for the secret key for sets of $S_1, S_2, ..., S_{q1}$.
3. At some point, the adversary $\mathcal{A}$ sends the challenger $\mathcal{B}$ two message M_0, M_1 on which it wishes to be challenged. In addition the adversary $\mathcal{A}$ gives a challenge access structure $\mathbb{A}$. The only restriction is that such that none of the sets $S_1, S_2, ..., S_{q1}$ from phase (2) satisfy the access structure. The challenger $\mathcal{B}$ picks a random $b \in \{0, 1\}$ and gives the adversary the challenge ciphertext C_{id}^*.
4. Adversary $\mathcal{A}$ can continue to ask for secret SK for any sets with the restriction that none of the attributes $S_{q1+1}, S_{q1+2}, ..., S_q$ satisfy the challenge access structure.
5. Eventually, the adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and wins the game if $b = b'$

The advantage of an adversary is defined to be $Adv = |Pr[b = b'] - 1/2|$ in the game.

Definition 2. *(IND-CPA) DMU-ABSE is semantically secure against an adaptive chosen plaintext attack if all probabilistic polynomial-time (PPT) attackers have at most negligible advantage in λ in the above security game.*

3 Preliminaries

In this section, we will introduce some background on bilinear maps and generic group model.

3.1 Access Structure

There are several kinds of access structure in ABE scheme, such as threshold structure, tree-based structure and AND-gate structure. In our construction, we adopt a series of AND-gate on multi-value attribute as our access structure. It is assumed that the total number of attributes set is n, and all n attributes be indexed as $U = att_1, att_2, ..., att_n$. For every attribute $att_i \in U, (i = 1, 2, ..., n)$, $V_i = v_{i,1}, v_{i,2}, ..., v_{i,n_i}$ be a set of possible values of att_i, where n_i is the number of the possible values for att_i. Each user is defined as an attribute list $S = (x_1, x_2, ..., x_n)$, where $x_i \in V_i$. The access structure in ciphertext is defined as $\mathbb{A} = (W_1, W_2, ..., W_n)$, where $W_i \in V_i$. The attribute list S will satisfy the access structure $\mathbb{A}$ if and only if $x_i = W_i, (i = 1, 2, ..., n)$.

3.2 Bilinear Maps

A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

1. Bilinearity: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.
3. Computability: $e(u, v)$ is efficiently computable for any $u, v \in G$.

3.3 Generic Group Model

Let $\Upsilon = (p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow Pair(1^\lambda)$. It is defined by [2] that: considering three random encodings $\psi_0, \psi_1 : \mathbb{Z}_p \rightarrow \{0,1\}^m$, where $m > 3log(p)$. Let $\mathbb{G} = \{\psi_0(x) | x \in \mathbb{Z}_p\}$ and $\mathbb{G}_T = \{\psi_1(x) | x \in \mathbb{Z}_p\}$. Oracles are used to compute the induced group action on $\mathbb{G}, \mathbb{G}_T$ and compute a non-degenerate multilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Random oracles are also given to present the hash functions H_1, H_2, H_3.

4 DMU-ABSE

4.1 Construction

1. **Setup**$(1^\lambda) \rightarrow PK, MSK, MPK$
Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of order p, and let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be the bilinear map. The algorithm randomly chooses $\alpha, \beta, \nu \in \mathbb{Z}_p$, $g, h \in \mathbb{G}$ and three hash functions H_1: $\{0,1\}^* \rightarrow \mathbb{G}$, H_2: $\{0,1\}^* \rightarrow \mathbb{Z}_p$, H_3: $\{0,1\}^* \rightarrow \{0,1\}^\lambda$. let $A = e(g,g)^\alpha$, $B = g^\beta$, $V = h^\nu$. The system generates public key $PK = (\gamma, g, A, B, H_1, H_2, H_3)$, the master key $MSK = (\alpha, \beta, \nu)$, and the master proxy key $MPK = V$. MPK is send to the proxy server only.
2. **Enroll**$(MSK, ID) \rightarrow EK_{ID}, RK_{ID}$
Trusted authority randomly chooses a number $r_{ID} \in \mathbb{Z}_p$ for the user ID and calculates $RK_{ID} = g^{\nu/r_{ID}}, M_1 = g^{r_{ID}}, M_2 = h^{r_{ID}}$. Finally, the enroll key $EK_{ID} = (M_1, M_2)$ is send to user ID, (ID, RK_{ID}) is send to proxy server.
3. **KeyGenUser**$(MSK, S) \rightarrow SK$
Trusted authority takes input as a set of attributes $S = (x_1, x_2, ...x_n)$ and selects random $r, r_i \in \mathbb{Z}_p (i = 1, 2, ..., n)$, set $r_s = \sum_{i=1}^{n} r_i$. For each attribute $x_i \in S$, trusted authority calculates $\hat{D}_i = g^{r_i} \cdot H_1(x_i)^r$, $D_1 = g^{(\alpha + r_s)/\beta}$, $D_2 = \prod_{i=1}^{n} H_1(x_i)^\beta$, $D_3 = g^r$. The search key is send to data user as

$$<SK = (D_1, D_2, D_3, \{\hat{D}_i\} | x_i \in S)>$$

4. **Encrypt**$(PK, M \in \mathbb{G}_2, EK_{ID}, W, \mathbb{A}) \rightarrow C_w$
Given a decryption key of the file $M \in \mathbb{G}_1$ and an AND gate policy $\mathbb{A} = (W_1, W_2 ..., W_n)$, and keywords list $W = (w_1, w_2, ..., w_m)$ in the file id, data owner selects random number $s_1, s_2 \in \mathbb{Z}_p^*$ and calculate $C_1 = B^{s_1}$, $C_2 = g^{s_1}$, $C_3 = M \cdot A^{s_1}$, $\hat{C}_i = H_1(W_i)^{s_1}$, $C_4^k = M_1^{H_2(w_k)} \cdot M_2^{s_2}$, where $w_k \in (w_1, w_2, ...w_m)$, $C_5 = g^{s_2}$. Finally, encrypted file id and all these encrypted indexes will be send to proxy server.

$$<\{C_{w_k}\}_{k=1,...,m} | \ C_{w_k} = (C_1, C_2, C_3, C_4^k, C_5, \hat{C}_i)>$$

5. **Re-Encrypt**$(MPK, C_w, RK_{ID}) \rightarrow C'_w$
For every $\{C_{w_k}\}_{k=1,...,m}$, the proxy server calculates the re-encrypted label L_p^k and sends the re-encrypted indexes with corresponding encrypted file to the cloud server:

$$<\{C'_{w_k}\}_{k=1,...,m} | \ C'_{w_k} = (C_1, C_2, C_3, \hat{C}_i, L_p^k), L_p^k = H_3(\frac{e(C_4^k, RK_{ID})}{e(C_5, V)})>$$

6. **MatrixBuild**$(C'_w, id, \mathbb{M}) \rightarrow \mathbb{M}'$
The cloud server firstly stores the file id and marks its address as A_{id}. Then it queries the look-up table $\mathbb{T}$ for every label L_p^k in $\{C'_{w_k}\}$:
   (a) If $\mathbb{T}[L_p^k] = NULL$, create a linked list $\mathbb{L}$ and sets the value of $\mathbb{T}[L_p^k]$ as the pointer of the list $\mathbb{L}$;
   (b) If $\mathbb{T}[L_p^k] \neq NULL$, set $\mathbb{L}$ as the value of $\mathbb{T}[L_p^k]$;

(c) Finally it creates a new index block value $< C_{id}, A_{id}, tagbit >$, where $C_{id} = (C_1, C_2, C_3, \hat{C}_i)$, and $tagbit = 0$ and adds it to the end of the linked list $\mathbb{L}$.
(d) After all $\{C'_{w_k}\}$ have been inserted, it creates a file array list $\mathbb{L}_f^{id}$ indexed by the file id and adds all the blocks above into the list.

7. **Trapdoor** $(SK, EK_{ID}, w) \rightarrow t_w$
Data user randomly selects $t \in \mathbb{Z}_p^*$ and calculates $tk_1 = D_2^t$, $tk_2 = B^t$, $tk_3 = M_1^{H_2(w)} \cdot M_2^t$, $tk_4 = g^t$, then sets the trapdoor of the chosen keyword w as:
$$<t_w = (tk_1, tk_2, tk_3, tk_4)>$$
8. **Re-Trapdoor**$(MPK, t_w, RK_{ID}) \rightarrow t'_w$
Proxy server calculates re-encrypted label L_p, and sends the re-encrypted trapdoor t'_w to the cloud server:
$$<t'_w = (tk_1, tk_2, L_p), L_p = H_3(\frac{e(tk_3, RK_{ID})}{e(tk_4, V)})>$$
9. **Search**$(t'_w, \mathbb{M}) \rightarrow \mathcal{R}$
The cloud server searches for L_p in the look-up table, sets $P_w = \mathbb{T}[L_p]$ and creates a new empty result list $\mathcal{R}$, then follows:
(a) If $P_w = NULL$, return $\mathcal{R}$ and abort;
(b) If $tagbit = 1$, delete this linked list block and go to step (d);
(c) Retrieve index ciphertext $C_{id} = (C_1, C_2, C_3, \hat{C}_i)$ in P_w and run the **Test Equation** $e(tk_1, C_2) = e(\prod_{i=1}^{n} \hat{C}_i, tk_2)$ to check whether data user's attributes satisfy the access policy embed in the ciphertext. If the equation holds, add P_w to $\mathcal{R}$;
(d) Set $P_w =$ next pointer of P_w and go to step (a);
After searching, cloud server finds out all encrypted files according to the A_{id} in the list $\mathcal{R}$, and returns them with corresponding index ciphertext C_{id} in $\mathcal{R}$ to the data user.
10. **Decrypt**$(SK, C_{id}) \rightarrow M$
After receiving the encrypted files and corresponding encrypted ciphertext C_{id}, data user can calculate E if his attribute list S satisfies the access policy $\mathbb{A}$, i.e. $x_i = W_i (i = 1, 2, ..., n)$.
$$E = \prod_{i=1}^{n} \frac{e(\hat{D}_i, C_2)}{e(D_3, \hat{C}_i)} = e(g, g)^{r_s s_1}$$
Then data user can get the corresponding decryption key M of the encrypted file by calculating
$$M = \frac{C_3}{(e(D_1, C_1)/E)}$$
11. **FileDeletion**$(id, \mathbb{M})$
The cloud server takes file id to find the file list $\mathbb{L}_f^{id}$ in the index matrix $\mathbb{M}$, and alters the tag bits from '0' to '1' in the index blocks in the list $\mathbb{L}_f^{id}$.

12. **Revocation**(ID)
The trusted authority uses ID to instruct the proxy server to delete the corresponding tuple (ID, RK_{ID}) in the proxy server. Without RK_{ID}, user's indexes and trapdoors won't be re-encrypted by the proxy server.

4.2 Correctness

Correctness (PRE): We now show that the **Re-encryption** and **Re-trapdoor** generate same labels of the same keyword:

$$L_p = H_3(\frac{e(C_4, RK)}{e(C_5, V)}) = H_3(\frac{e(M_1^{H_2(w)} \cdot M_2^{s_2}, g^{\nu/r_{ID}})}{e(g^{s_2}, h^{\nu})})$$
$$= H_3(\frac{e(g^{r_{ID}H_2(w)} \cdot h^{r_{ID}s_2}, g^{\nu/r_{ID}})}{e(g^{s_2}, h^{\nu})}) = H_3(e(g,g)^{H_2(w)\nu}) = H_3(\frac{e(tk_3, RK)}{e(tk_4, V)})$$

Correctness (Search): We now show the correctness of the **Test Equation** in the **Search** phase:

$$e(tk_1, C_2) = e(D_2^t, g^{s_1}) = e(\prod_{i=1}^{n} H(x_i)^{\beta t}, g^{s_1}) = e(\prod_{i=1}^{n} \hat{C}_i, tk_2)$$

Correctness (Decrypt): We now show the correctness of the **Decrypt** phase:

$$\frac{C_3}{(e(D_1, C_1)/E)} = \frac{C_3}{e(g^{\alpha + r_s/\beta}, g^{\beta s_1})/e(g.g)^{r_s s_1}} = \frac{M \cdot e_o(g,g)^{\alpha s_1}}{e(g.g)^{\alpha s_1}} = M$$

5 Security Analysis

In this section, we will give the security analysis of our dynamic multi-user attribute-based searchable encryption system proposed in Sect. 4.

We now prove that DMU-ABSE is IND-CKA secure in the generic bilinear group model.

Theorem 1. *Let $\gamma = (p, \mathbb{G}, \mathbb{G}_T, e)$ be defined as above. For any adversary $\mathcal{A}$ let q be a bound on the total number of group elements it receives from queries it makes to the oracles for the hash functions, groups $\mathbb{G}, \mathbb{G}_T$, the bilinear map e, and from its interaction with the IND-CKA security game. Then we have that the advantage of the adversary in the IND-CKA security game is negligible in λ.*

Proof. we initialize $g = \psi_0(1), g_T = \psi_1(1)$. We will write $g^x = \psi_0(x)$, $e(g,g)^y = \psi_1(y)$. In the following queries, the adversary $\mathcal{A}$ will communicate with the simulator $\mathcal{B}$ using the ψ-representations of the group elements. $\mathcal{B}$ interacts with $\mathcal{A}$ in the security game as follows:

1. The challenger $\mathcal{B}$ randomly selects $\alpha, \beta \in \mathbb{Z}_p$, $g, h \in \mathbb{G}$ and set public key as $A = e(g,g)^{\alpha}, B = g^{\beta}, V = h^{\nu}$, and sends the public key to adversary $\mathcal{A}$. When the adversary (or simulation) calls for the evaluation of H_1, H_2, H_3 on any string x_i, a new random value t_i is chosen from $\mathbb{Z}_p$ (unless it has already been

chosen), and the simulation provides g^{t_i} as the response to $H_1(x_i), H_2(x_i)$ and $k_i = \{0,1\}^\lambda$ as the response to $H_3(x_i)$.

2. On $\mathcal{A}$'s secret key query for keyword w and user ID, a new random value $r^{(j)}$ is chosen from $\mathbb{Z}_p$, and for every attribute $x_j \in S_j$, new random values $r_i^{(j)}$, r'_j are chosen from $\mathbb{Z}_p$. The simulator $\mathcal{B}$ computes: $D_1 = g^{\frac{\alpha + r^{(j)}}{\beta}}$, $D_2 = \prod_{i=1}^{n} g^{t_i^{(j)}\beta}$, $D_3 = g^{r'_j}$, and for each attribute $x_i \in S_i, i = (1,2,...,n)$, $\mathcal{B}$ sets: $\hat{D}_i = g^{r_i^{(j)} + t_i^{(j)} r'_j}$. The secret key is defined as $SK = (D_1, D_2, D_3, \{\hat{D}_i\})$. Finally, $\mathcal{B}$ randomly chooses t_k, r_k from $\mathbb{Z}_p$ and generates the trapdoor of the chosen keyword w by: $tk_1 = D_2^{t_k}$, $tk_2 = B_k^t$, $tk_3 = g^{H_2(w) r_k} h^{r_k t_k}$, $tk_4 = g_k^t$ and then re-encrypts t_w by: $L_p = H_3(\frac{e(tk_3, g^{\nu/r_k})}{e(tk_4, h^\nu)}) = H_3(e(g,g)^{H_2(w)v})$. Then simulator $\mathcal{B}$ sends the re-encrypted trapdoor $t'_w = (tk_1, tk_2, L_p)$ to adversary $\mathcal{A}$.
3. Eventually adversary $\mathcal{A}$ produces a pair of keyword w_0 and w_1 that it wishes to be challenged on. The challenger $\mathcal{B}$ picks a random $b \in \{0,1\}$ and creates the challenging trapdoor as follows: $tk_1 = D_2^{t_k}$, $tk_2 = B^{t_k}$, $L_{p_b} = H_3(e(g,g)^{H_2(w_b)v})$. Finally, simulator $\mathcal{B}$ sends $t'_{w_b} = (tk_1, tk_2, L_{p_b})$ to adversary $\mathcal{A}$.
4. $\mathcal{A}$ repeats the query of phase (2) with the restriction that $\mathcal{A}$ did not previously ask for the trapdoors t'_{w_0}, t'_{w_1}.
5. Eventually, the adversary $\mathcal{A}$ outputs $b' \in \{0,1\}$

We can instead consider a modified game in which the real challenging ciphertext via substituting $H_3(e(g,g)^{H_2(w_b)v})$ for $H_2(e(g,g)^\theta)$. The probability for distinguishing $H_3(e(g,g)^{H_2(w_0)v})$ from $e(g,g)^\theta$ is equal to half of the probability for distinguishing $H_3(e(g,g)^{H_2(w_0)v})$ from $H_3(e(g,g)^{H_2(w_1)v})$.

Suppose hash functions H_2 and H_3 are respectively modeled as two random oracles, and random value $k_i = \{0,1\}^\lambda$ is the response of $H_3(x_i)$. Then the ideal game for adversary $\mathcal{A}$ is to distinguish two random values k_1 and k_2 which are randomly choose from $\{0,1\}^\lambda$. Obliviously the probability for distinguish k_1 and k_2 is 0 unless there is a collision of H_3.

The probability that "unexpected collision" occurs is at most $O(q^2/p)$ before substitution by the Schwartz-Zipple lemma [13]. On the other hand, even if the adversary could distinguish $e(g,g)^{H_2(w_0)v}$ and $e(g,g)^\theta$, he is still unable to distinguish between $H_3(e(g,g)^{H_2(w_0)v})$ and $H_3(e(g,g)^\theta)$.

Therefore, we can conclude that $\mathcal{A}$ gains no unnegligible advantage in the modified game, which means that $\mathcal{A}$ gains a negligible advantage in the IND-CKA game. This completes the proof.

We now prove that DMU-ABSE is IND-CPA secure in the generic bilinear group model.

Theorem 2. *Let $\gamma = (p, \mathbb{G}, \mathbb{G}_T, e)$ be defined as above. For any adversary $\mathcal{A}$ let q be a bound on the total number of group elements it receives from queries it makes to the oracles for the hash functions, groups $\mathbb{G}, \mathbb{G}_T$, the bilinear map e, and from its interaction with the IND-CPA security game. Then we have that the advantage of the adversary in the IND-CPA security game is $O(q^2/p)$.*

Proof. we initialize $g = \psi_0(1), g_T = \psi_1(1)$. We will write $g^x = \psi_0(x)$, $e(g,g)^y = \psi_1(y)$. In the following queries, the adversary $\mathcal{A}$ will communicate with the simulator $\mathcal{B}$ using the ψ-representations of the group elements. $\mathcal{B}$ interacts with $\mathcal{A}$ in the security game as follows:

1. The challenger $\mathcal{B}$ randomly selects $\alpha, \beta \in \mathbb{Z}_p$, and set public key as $A = e(g,g)^{\alpha}$, $B = g^{\beta}$, and sends the public key to adversary $\mathcal{A}$. When the adversary (or simulation) calls for the evaluation of H_1, H_2 on any string x_i, a new random value t_i is chosen from $\mathbb{Z}_p$ (unless it has already been chosen), and the simulation provides g^{t_i} as the response to $H_1(x_i)$, $H_2(x_i)$.
2. On $\mathcal{A}$'s secret key query for set $S_j = \{x_1, x_2, ..., x_n\}$, a new random value $r^{(j)}$ is chosen from $\mathbb{Z}_p$, and for every attribute $x_j \in S_j$, new random values $r_i^{(j)}$, r'_j are chosen from $\mathbb{Z}_p$. The simulator $\mathcal{B}$ computes: $D_1 = g^{\frac{\alpha + r^{(j)}}{\beta}}$, $D_2 = \prod_{i=1}^{n} g^{t_i^{(j)}\beta}$, $D_3 = g^{r'_j}$, and for each attribute $x_i \in S_i, i = (1, 2, ..., n)$, $\mathcal{B}$ sets: $\hat{D}_i = g^{r_i^{(j)} + t_i^{(j)} r'_j}$. The secret key is defined as $SK = (D_1, D_2, D_3, \{\hat{D}_i\})$ Then send the secret key SK to adversary $\mathcal{A}$.
3. Eventually adversary $\mathcal{A}$ produces a pair of message M_0 and M_1 and a challenge access structure $\mathbb{A}$. The challenger $\mathcal{B}$ selects random number $s_1 \in \mathbb{Z}_p^*$, picks a random $b \in \{0,1\}$ and sets $C_1 = g^{\beta s_1}$, $C_2 = g^{s_1}$, $C_3 = M_b \cdot e(g,g)^{\alpha s_1}$. For each attribute in the AND gate $\mathbb{A}$, let $\hat{C}_i = g^{t_i s_1}, i = (1, 2, ..., n)$. These values are sent to $\mathcal{A}$.
4. $\mathcal{A}$ repeats the query of phase (2) with the restriction that $\mathcal{A}$ did not previously ask for the secret key for the attribute set S_i which satisfy the challenge access structure $\mathbb{A}$.
5. Eventually, the adversary $\mathcal{A}$ outputs $b' \in \{0,1\}$

We can instead consider a modified game in which the real challenging ciphertext via substituting $M_b \cdot e(g,g)^{\alpha s_1}$ for $e(g,g)^{\theta}$. The probability for distinguishing $M_0 \cdot e(g,g)^{\alpha s_1}$ from $e(g,g)^{\theta}$ is equal to half of the probability for distinguishing $M_0 \cdot e(g,g)^{\alpha s_1}$ from $M_1 \cdot e(g,g)^{\alpha s_1}$.

We suppose that $\mathcal{B}$' simulation is prefect as long as no "unexpected collision" happens. More precisely, we think of an oracle query as being a rational function $\nu = \eta/\xi$ in the variables θ, α, β, $t_i^{(j)}$, $t^{(j)}$, $r_i^{(j)}$, r'_j, $r^{(j)}$, s_1, s_2. An unexpected collision would be when two queries corresponding to two distinct formal rational functions $\eta/\xi = \eta'/\xi'$ but where due to the random choices of these variables values, we have that the values of $\eta/\xi = \eta'/\xi'$.

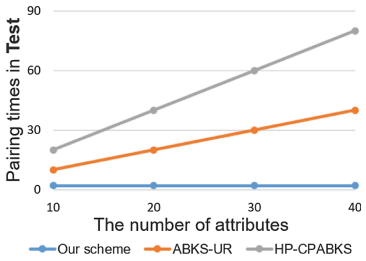
The probability that "unexpected collision" occurs is at most $O(q^2/p)$ before substitution by the Schwartz-Zipple lemma [13]. The adversary's view would have been identically distributed even if $\mathcal{B}$ substitutes αs_1 for variable θ. Since θ only occurs as $e(g,g)^{\theta}$, we must have that $\nu - \nu' = \gamma\alpha s_1 - \gamma'\theta$. The adversary can almost never construct a query for $e(g,g)^{\gamma\alpha s_1}$. To construct the term αs_1, the adversary can pairing $s_1\beta$ with $(\alpha + r^{(j)})/\beta$. In this way, $\mathcal{A}$ must create a query polynomial containing $\gamma\alpha s_1 + \Sigma\gamma' s_1 r^{(j)}$. In order to obtain a query of form αs_1, $\mathcal{A}$ must cancel the terms of form $\Sigma\gamma' s_1 r^{(j)}$.

To construct the term αs_1, the adversary can pairing $s_1\beta$ with $(\alpha + r^{(j)})/\beta$. But according to the simulation, $\mathcal{A}$ cannot get the secret key. Thus $\mathcal{A}$ is unable to get the form of αs_1 and construct the query for $e(g,g)^{\gamma\alpha s_1}$. By the

Schwartz-Zipple lemma, we can conclude that $\mathcal{A}$ gains a negligible advantage in the modified game, which means that $\mathcal{A}$ gains no unnegligible advantage in the IND-CPA game. This completes the proof.

	[5]	[10]	[12]	[16]	[17]	Our scheme
Fine-grained access control	√	√	√	√	√	√
Constant trapdoor	√	x	x	x	√	√
Multi-user	√	x	x	x	x	√
Hidden policy	x	√	√	x	√	√
File decryption	x	√	x	x	√	√
Dynamic update	x	x	x	x	x	√

(a) Function comparison



(b)Pairing times

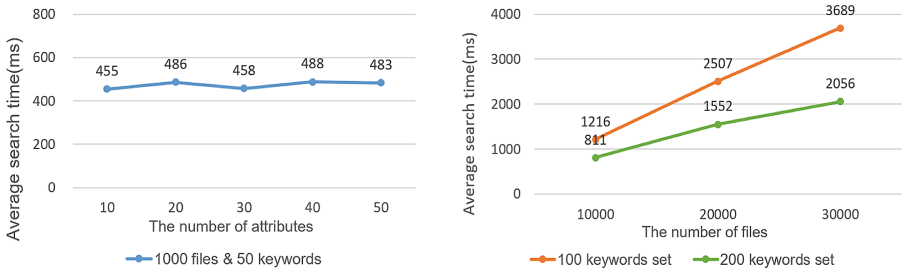
Fig. 3. Performance comparison with other related works

6 Comparison and Experiments

Functionality Comparisons: We list the key features of our scheme in Fig. 3(a) and make a comparison with several schemes, including MUSE-CK [5], KSF-OABE [10], HP-CPABKS [12], ABKS-UR [16], and CP-ABKS [17], in terms of supporting multi-user, constant-size trapdoor, hidden policy, file decryption and dynamic operation. Among these schemes, [5,10,12] and [17] are based on AND gate access structure. From the comparison, we can see that only [5] and our scheme achieve multi-user setting. Only in [5,17] and our scheme, the size of trapdoor are non-linear with the number of attributes involved in the access policy. Besides, only our scheme and [17] support decryption of the encrypted files in the search result. In particular, our multi-user attribute-based searchable encryption system also supports dynamic operations on the cloud including file updating and deleting.

Figure 3(b) demonstrates the comparison of pairing times of one search overhead between HP-CPABKS [12], ABKS-UR [16] and our scheme. It shows the number of pairing operations of our scheme does not change with the number of attributes during encryption, while the number of pairing operation of [12] and [16] are linear to the number of attributes. This makes our scheme more efficient in the complicated attribute scenarios.

Experimental Evaluation: In order to show the efficiency of our system, we conduct experiments with JPBC library on java8, which are executed on an AMD Ryzen5 2500U at 2.0 GHz and 8 GB memory. We exploit the Type A pairings conducted on the curve $y^2 = x^3 + x$ over the field $\mathbb{Z}_p$ for some prime $p = 3 \pmod 4$. Experiments in Fig. 4(a) shows the relation between search time and the number of attributes in archives of 1000 files. It can be seen that the number of attributes barely influences the search time, which verifies that our scheme is independent of the number of attributes. Then in the scenario that the size of



(a) Different size of attributes set (b) Different size of keywords set

Fig. 4. Experimental results of average search time

the attribute set is 50, Fig. 4(b) shows the relation between search time and the number of files. It indicates that even in a huge archive, the search time will be reduced if the keywords set is increased, which implies our system is efficient in practical scenarios.

7 Conclusion

In this paper, a dynamic multi-user ciphertext-policy attribute-based searchable encryption system, DMU-ABSE, is presented. In DMU-ABSE, data owners authorize data users in a fine-grained manner by specifying access policy in the index ciphertexts. Meanwhile, DMU-ABSE achieves hidden-policy, constant size and non-deterministic properties. With the collaboration of the proxy server, the indexes and trapdoors will be re-encrypted before being sent to the cloud server so that the cloud server can merge them into one index matrix and search within it without learning any information. Furthermore, DMU-ABSE supports dynamic archives and user revocation, which is more practical in the scenario of PHR and online subscription systems. A concrete and formal proof is given to show that the proposed scheme is IND-CKA and IND-CPA secure in the sense of semantic security. The performance analysis demonstrates that the proposed scheme is efficient and practical.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China (Grant No. 61632012, 61672239, 61602180 and U1509219), in part by Natural Science Foundation of Shanghai (Grant No. 16ZR1409200), and in part by "the Fundamental Research Funds for the Central Universities". Zhenfu Cao and Jiachen Shen are the corresponding authors.

References

1. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_30

2. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_26
3. Cao, Z.: New Directions of Modern Cryptography. CRC Press, Boca Raton (2012)
4. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_30
5. Chang, Y.J., Wu, J.L.: Multi-user searchable encryption scheme with constant-size keys. In: 2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2), pp. 98–103. IEEE (2017)
6. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. J. Comput. Secur. **19**(5), 895–934 (2011)
7. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98. ACM (2006)
8. Hahn, F., Kerschbaum, F.: Searchable encryption with secure and efficient updates. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 310–320. ACM (2014)
9. Khader, D.: Introduction to attribute based searchable encryption. In: De Decker, B., Zúquete, A. (eds.) CMS 2014. LNCS, vol. 8735, pp. 131–135. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44885-4_11
10. Li, J., Lin, X., Zhang, Y., Han, J.: KSF-OABE: outsourced attribute-based encryption with keyword search function for cloud storage. IEEE Trans. Serv. Comput. **10**(5), 715–725 (2017)
11. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 111–129. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68914-0_7
12. Qiu, S., Liu, J., Shi, Y., Zhang, R.: Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack. Sci. China Inf. Sci. **60**(5), 052105 (2017)
13. Schwartz, J.: Fast polynomial algorithms for verification of polynomial identities. J. Assoc. Comput. **27**(4), 701–717 (1980)
14. Shao, J., Cao, Z.: CCA-Secure proxy re-encryption without pairings. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00468-1_20
15. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE Symposium on Security and Privacy, S&P 2000, pp. 44–55. IEEE (2000)
16. Sun, W., Yu, S., Lou, W., Hou, Y.T., Li, H.: Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. IEEE Trans. Parallel Distrib. Syst. **27**(4), 1187–1198 (2016)
17. Wang, H., Dong, X., Cao, Z.: Multi-value-independent ciphertext-policy attribute based encryption with fast keyword search. IEEE Trans. Serv. Comput. (2017). https://doi.org/10.1109/TSC.2017.2753231
18. Zheng, Q., Xu, S., Ateniese, G.: VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 522–530. IEEE (2014)