RESEARCH ARTICLE

A keyword searchable attribute-based encryption scheme with attribute update for cloud storage

Shangping Wang¹, Jian Ye^{1*}, Yaling Zhang²

1 School of Science, Xi'an University of Technology, Xi'an, Shaanxi, China, **2** School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, Shaanxi, China

* wayejian@163.com

OPEN ACCESS

Citation: Wang S, Ye J, Zhang Y (2018) A keyword searchable attribute-based encryption scheme with attribute update for cloud storage. PLoS ONE 13 (5): e0197318. <https://doi.org/10.1371/journal.pone.0197318>

Editor: Kim-Kwang Raymond Choo, University of Texas at San Antonio, UNITED STATES

Received: November 7, 2017

Accepted: April 29, 2018

Published: May 24, 2018

Copyright: © 2018 Wang et al. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: This research is supported by the National Natural Science Foundation of China (No. 61572019, 61173192, <http://www.nsf.gov.cn/>), the Key Project of Research Foundation of Natural Science Foundation of Shaanxi Province of China (NO. 2016JZ001, <http://www.sninfo.gov.cn/>), and the Key Laboratory Research Project of Education Bureau of Shaanxi Province of China (No. 16JS078, <http://www.snedu.gov.cn/>). The funders

Abstract

Ciphertext-policy attribute-based encryption (CP-ABE) scheme is a new type of data encryption primitive, which is very suitable for data cloud storage for its fine-grained access control. Keyword-based searchable encryption scheme enables users to quickly find interesting data stored in the cloud server without revealing any information of the searched keywords. In this work, we provide a keyword searchable attribute-based encryption scheme with attribute update for cloud storage, which is a combination of attribute-based encryption scheme and keyword searchable encryption scheme. The new scheme supports the user's attribute update, especially in our new scheme when a user's attribute need to be updated, only the user's secret key related with the attribute need to be updated, while other user's secret key and the ciphertexts related with this attribute need not to be updated with the help of the cloud server. In addition, we outsource the operation with high computation cost to cloud server to reduce the user's computational burden. Moreover, our scheme is proven to be semantic security against chosen ciphertext-policy and chosen plaintext attack in the general bilinear group model. And our scheme is also proven to be semantic security against chosen keyword attack under bilinear Diffie-Hellman (BDH) assumption.

Introduction

Attribute-based encryption (ABE)[1–4] is regarded as an effective encryption method with fine grained access control in the cloud storage. Attribute-based encryption can be divided into two types of key-policy attribute-based encryption [1] (KP-ABE) and ciphertext-policy attribute-based encryption [2] (CP-ABE). The KP-ABE scheme refers to that the ciphertext is associated with an attribute set, and a user's secret key is associated with an access policy. A user can decrypt the ciphertext if and only if the ciphertext's attribute set satisfy the access policy of user's secret key. The CP-ABE scheme refers to that the ciphertext is associated with an access policy, and a user's secret key is associated with an attribute set. A user is can decrypt the ciphertext if and only if his attribute set satisfy the access policy of the ciphertext.

At present, many ABE schemes [5–9] have been proposed, which provide secure data access control and overcome the shortcomings of one-to-one encryption pattern in identity-based encryption scheme. However, these schemes are still defective to be used in practice, as the

had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

attribute of a user is dynamic, which may be changed over time. Thus the attribute revocation mechanism is necessary for ABE scheme to be used in practice.

The revocation mechanism can be divided into two types: direct revocation mechanism and indirect revocation mechanism. Imai and Attrapaduang [10] gives a clear definition of direct revocation and indirect revocation. Direct revocation is defined as: the sender specifies a revocation list when encrypting the data. Indirect revocation is defined as: the authorized institutions regularly issue key updates to non-revoked users. At present, many schemes with direct revocation [11–14] have been proposed. Li et al. [11] proposed an identity-based revocation scheme that performs directed revocation by giving the revocation rights to encipherer directly. Tu et al. [14] proposed a revocable ABE scheme. In addition, some indirect attribute revocation schemes [15–18] have also been proposed. Yu et al. [15] proposed an attribute based data sharing scheme with attribute revocation. In this scheme, user's any attribute can be revoked by proxy re-encryption technique. Li et al. [18] proposed a scheme that supports user's attribute revocation, but the scheme could only revoke a single attribute of the user, thus it could not satisfy the actual needs.

The attribute update is another significant problem in the ABE environment. In actual life, a user's attribute set may need to be updated over time when his working role may be changed. For example, assume that Alice is a company employee, then her attribute set needs to be updated when her working role is promoted from a programmer lifted to a project manager, thus her former attribute set $A = \text{"female, programmer"}$ should be changed to a new attribute set $B = \text{"female, project manager"}$. And the attribute authority (AA) should issue an update key to update Alice's secret key. Meanwhile, the attribute authority must ensure that the employee Alice cannot further use her previous key related to the attribute set "female, programmer" to access the ciphertexts. Thus, the attribute update is not a simple process. Some attribute update schemes [19–21] have been proposed. However, these schemes have a common problem, the problem is that if there is a user's an attribute is updated, and then many other user's secret key and a lot of ciphertexts related with this attribute need to be updated, which will undoubtedly waste a lot of computational resources.

To address this problem, we give a feasible solution in this paper. The main idea of our solution is that the secret key of a user is divided into two parts, one part which is irrelevant to attribute is retained by the user, and the other part which is relevant to attribute is sent to the cloud server (CS). When an attribute of any user needs to be updated, the AA issues an update key to CS. Then CS only updates the secret key of this attribute for all valid users, and other secret key of all user and the ciphertexts related with this attribute need not to be updated. This method will greatly reduce the work load of the system.

Although attribute based encryption technology provides an effective means for data confidentiality, yet it brings another new problem that the users may find it difficult to search for interesting data from a vast number of encrypted data. This problem is called keyword search problem [22]. One of the simplest searching methods is to download all encrypted data locally and then to decrypt it, finally to execute keyword search in plaintext. However, this method will waste huge computational resource and bring a vast cost for user to do the work of decryption.

Another extreme searching method is to send the secret key of the user and keywords to CS, then CS decrypts all of the ciphertexts and performs searching operation on plaintext. But this method will expose the user's secret key and privacy of search keyword to CS, this is infeasible. Some search-based encryption schemes [23–26] have been proposed. Such as Boneh et al. [23] first proposed a public key encryption with keyword search scheme. Dan and Ostrovsky [24] proposed a public key cryptographic scheme that allows privacy data retrieval (PIR), and allows multiple data contributors to upload their data with public key by encryption algorithm, and only the user with the corresponding secret key can decrypt the data.

Some search encryption schemes [27–30] focuses on search efficiency have also been proposed. Fu et al. [27] proposed a scheme that not only supports multi-keyword ranked search but also provides parallel search. Li et al. [28] put forward a scheme which supports multi-keyword search. In this scheme, users can retrieve multiple keywords at once, and which greatly improves the search efficiency and search accuracy. Sun et al. [30] proposed a verifiable attribute-based keyword search scheme that supports fine-grained search authorization scheme. In this scheme, multiple data owners and multiple users are supported, and the scheme also supports fine-grained search authorization.

In addition, some schemes focus on achieving both attribute revocation and keyword search have been proposed [31,32]. The schemes [31,32] which not only support user's multiple attributes revocation but also provide keyword search. However, our new scheme is different from the schemes [31,32], and the differences between our scheme and schemes [31,32] can be described as follows: firstly, the scheme [32] is based on the key-policy (KP-ABE). The new scheme and the scheme [31] are based on the ciphertext policy (CP-ABE), where the scheme [31] makes use of the access tree as access policy, while the new scheme makes use of LSSS as access policy. So the new scheme is different from the schemes [31,32]. Secondly, the scheme [31] supports public keyword searchable, and the keyword index and trapdoor are generated with the help of the cloud server. The ciphertext and the keyword index of the scheme [32] are associated with attribute. The new scheme also supports public key keyword search, but the keyword index and trapdoor generation phase is independently realized by user. Furthermore, the based on the difficult problems of the new scheme and schemes [31,32] are different. The scheme [31] is proven to be secure under the assumption of bilinear Diffie-Hellman (BDH) in selective security model. The scheme [32] is proven to be secure under the assumption of decisional bilinear Diffie-Hellman exponent (q-BDHE) and decisional Diffie-Hellman (DDH) in the selective security model. The new scheme is proven to be secure in the general bilinear group model.

Our contributions

In this paper, we propose a keyword searchable attribute-based encryption scheme with attribute update for cloud storage. The main contributions of our scheme are summarized as follows:

1. The new scheme is a combination of ABE scheme and keyword searchable encryption scheme. So our scheme not only solves the problem of confidentiality of the data with fine-grained access control but also solves the problem of keyword search. Moreover, the scheme is proven to be semantic security against chosen ciphertext-policy and chosen plaintext attack in the general bilinear group model.
2. The new scheme supports the user's attribute update, and when a user's attribute need to be updated, only the user's secret key related with this attribute need to be updated, while other users' secret key and the ciphertexts related with the attribute need not to be updated. This is a more efficient attribute update method than that in existing attribute update schemes.
3. In addition, the operation with high computation cost is outsourced to CS to reduce the user's computational burden.
4. Our keyword search algorithm supports multi-user keywords searchable, as long as user's trapdoor could match keywords index stored in the cloud storage. Moreover, our keyword search scheme is proved to be semantic security against chosen keyword attack (IND-CKA) under bilinear Diffie-Hellman (BDH) assumption.

Table 1. The comparisons of our scheme with some exiting schemes.

schemes	access control	secret key update for update user	No secret key update for non-update user	keyword searchable	No ciphertext update
[13]	LSSS	✓	✗	✗	✗
[19]	access tree	✓	✗	✗	✗
[21]	access tree	✓	✗	✗	✗
[29]	LSSS	✓	✗	✗	✗
[31]	access tree	✓	✗	✓	✗
Ours	LSSS	✓	✓	✓	✓

✗: there is no corresponding function in the scheme.

✓: there is corresponding function in the scheme.

<https://doi.org/10.1371/journal.pone.0197318.t001>

Functional comparisons

We compare the function of our scheme with some exiting schemes [13,19,21,29,31] in Table 1.

Preliminaries

Bilinear map [33]

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic bilinear groups of prime order p . Let g be a generator of $\mathbb{G}_0$. A bilinear map is a map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ with the following properties:

1. Bilinearity: for all $g \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(g^a, g^b) = e(g, g)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.
3. Computability: There is an efficient algorithm to compute $e(u, v)$ for $u, v \in \mathbb{G}_0$.

Bilinear Diffie-Hellman assumption [34]

The BDH problem in $\mathbb{G}_0$ is defined as follows: taken $(g, g^a, g^b, g^c) \in \mathbb{G}_0$ as input, compute $e(g, g)^{abc} \in \mathbb{G}_1$. We say that the adversary $\mathcal{A}$ has ϵ advantage in solving BDH problems in $\mathbb{G}_0$ if

$$\Pr[\mathcal{A}(g, g^a, g^b, g^c) = e(g, g)^{abc}] \geq \epsilon$$

We say that the BDH assumption holds in $\mathbb{G}_0$ if no probability polynomial adversary $\mathcal{A}$ has non-negligible advantage in solving the BDH problem in $\mathbb{G}_0$.

Generic bilinear group model [2]

We suppose there are two random encodings $\Psi_0, \Psi_1 : \mathbb{Z}_p^+ \rightarrow \{0, 1\}^*$, where $\mathbb{Z}_p$ is an additive group and $m > 3 \log p$. For $i = 0, 1$, we set $\mathbb{G}_i = \{\Psi_i(x) : x \in \mathbb{Z}_p^+\}$. We are given oracles to compute the induced group action on $\mathbb{G}_0, \mathbb{G}_1$ and an oracle to compute a non-degenerate bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. And we are also given a random oracle to represent the hash function H .

Linear secret sharing schemes [33]

A linear secret sharing scheme Π over a set of parties P is called linear (over $\mathbb{Z}_p$) if

1. The shares for each party form a vector over $\mathbb{Z}_p$.
2. There exists a matrix $\mathbf{M}$ with l rows and n columns called the share-generating matrix for Π . For all $i = 1, 2, \dots, l$, the function ρ defines the party labeling i th row of $\mathbf{M}$ as $\rho(i)$. When

we consider the column vector $v = (s, r_2, \dots, r_n) \in \mathbb{Z}_p^n$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \dots, r_n \in \mathbb{Z}_p$ are randomly chosen. Then Mv is the vector of 1shares of the secret s according to Π . The share $(Mv)_i$ belongs to party $\rho(i)$.

Suppose Π that is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and $I \subset \{1, \dots, l\}$. Then, there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π , then $\sum_{i \in I} \omega_i \lambda_i = s$. Furthermore, these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix M .

System model and security model

System model

A system framework of our scheme includes the main four entities is presented in Fig 1.

Attribute authority (AA). The AA is a perfectly trusted entity. It takes charge of the system establishment, user registration, attributes management and secret key generation. And when an attribute of a user needs to be updated, the AA generates an updated key for the user.

Cloud server (CS). The CS is responsible for storing the data and providing data access for legitimate users. It is also responsible for keyword search when a search trapdoor is received from a user. And it also takes charge of updating the user's partial secret key which related to the updated attribute and helps legitimate users to partially decrypt the ciphertext by using partial secret key of the user.

Fig 1. System model of the proposed scheme.

<https://doi.org/10.1371/journal.pone.0197318.g001>

Data owner (DO). The data owner encrypts its owner data and builds keyword indexes, and then outsources them to the CS.

User (U). Each legitimate user can search their interesting the files from system. The user generates a search trapdoor to protect the privacy of the search keyword. Then the user sends his identity and search trapdoor to CS. Without revealing any information about keyword search, the CS will find the encrypted file includes the keywords and do a lot of partial decryption work to reduce the decryption load of the user. Finally, the user gets the partial decrypted files, and then decrypts the partial decrypted files by using his owner partial secret key.

Algorithm description

We proposed a keyword searchable attribute-based encryption scheme with attribute update for cloud storage includes the following eight phases.

Phase 1: System initialization. $AA.Setup(\lambda, L) \rightarrow (PP, MSK, PK_s, SK_s)$. The setup algorithm inputs a security parameters λ and an attribute universe L , and outputs the public parameters PP , the master secret key MSK , the CS's public and secret key pairs (PK_s, SK_s) .

Phase 2: Key generation. $AA.KeyGen(MSK, id, S_{U_{id}}) \rightarrow (SK_{U_{id,1}}, SK_{U_{id,2}}, X_{U_{id}}, A_{priv}, B_{pub})$. The key generation algorithm inputs the master secret key MSK , an user's identity id and the user's attribute set $S_{U_{id}}$, and outputs the user's secret key $(SK_{U_{id,1}}, SK_{U_{id,2}}, X_{U_{id}})$, the user's search secret and public key pairs (A_{priv}, B_{pub}) .

Phase 3: File encryption and create keyword index. To get ciphertext $E_k(F)$, the DO encrypts file F with symmetric key k by the symmetric encryption algorithm. Then DO encrypts the symmetric key k by the following encryption algorithm.

DO.Encrypt $(PP, k, (M, \rho)) \rightarrow CT$. The encryption algorithm inputs the public parameter PP , the symmetric key k and the LSSS access structure (M, ρ) , and outputs a ciphertext CT .

DO.Index $(W, B_{pub}) \rightarrow I_W$. The index generation algorithm inputs a set of keywords W and data owner's search public key B_{pub} , and outputs the keywords index set I_W .

Phase 4: Trapdoor generation. $U.AuthorizationKey(PP, PK_s, X_{U_{id}}) \rightarrow K'_{U_{id}}$. The authentication information generation algorithm inputs public parameters PP , the CS's public key PK_s and the user's secret key $X_{U_{id}}$, and outputs the authentication information $K'_{U_{id}}$.

U.Trapdoor $(w, PK_s, A_{priv}) \rightarrow T_w$. The trapdoor generation algorithm inputs a keyword w , the CS's public key PK_s and the user's search secret key A_{priv} , and outputs the search trapdoor T_w .

Phase 5: Verification. $CS.Verifying(id, K'_{U_{id}}) \rightarrow (0, 1)$. The validation algorithm inputs user's identity id and the authentication information $K'_{U_{id}}$, and outputs 1 or 0.

Phase 6: File retrieval. $CS.Test(I_W, T_w) \rightarrow (0, 1)$. The test algorithm inputs the keywords index set I_W and the user's search trapdoor T_w , and outputs 1 or 0.

Phase 7: Data decryption. $CS.PreDecrypt(CT, SK_{U_{id,2}}) \rightarrow CT'$. The pre-decryption algorithm inputs the ciphertext CT for the access structure (M, ρ) and user's secret key $SK_{U_{id,2}}$ for the attribute set $S_{U_{id}}$. If the user's attribute set $S_{U_{id}}$ satisfies the access structure (M, ρ) . It outputs a partial decrypted ciphertext CT' . Otherwise, the algorithm is terminated.

U.PostDecrypt $(CT', SK_{U_{id,1}}) \rightarrow k$. The post-decryption algorithm inputs the partial decrypted ciphertext CT' and the user's secret key $SK_{U_{id,1}}$, and outputs symmetric key k .

Finally, the user decrypts the file $E_k(F)$ by the symmetric key k , and then the user gets the file F .

Phase 8: Attribute update. Assume that a user with identity id whose an attribute $j \in S_{U_{id}}$ needs to be updated to a new attribute j' by the AA. The attribute update phase includes five steps: (1) The AA executes update key algorithm to generate update key $UK_{j \rightarrow j'}$ and sends it to CS, and the AA informs CS that the user with identity id and his an attribute j will be updated

to a new attribute j' ; (2) The CS finds user's attribute set $S_{U_{id}}$ and secret key $SK_{U_{id},2}$ in the SL-list; (3) The CS updates the attribute j of the user to the attribute j' , and sets the new attribute set as $S_{U_{id}} := \{S_{U_{id}} \setminus \{j\}\} \cup \{j'\}$; (4) The CS also updates secret key $\{j : D_j, D'_j\}$ associates with attribute j to new secret key $\{j' : D_{j'}, D'_{j'}\}$ associates with attribute j' by update key $UK_{j \rightarrow j'}$; (5) The CS retains user's new attribute set $S_{U_{id}}$ and new secret key $SK'_{U_{id},2}$ in the SL-list.

AA.UKeyGen (PP, MSK, j, j') $\rightarrow UK_{j \rightarrow j'}$. The update key generation algorithm inputs the public parameter PP , the master secret key MSK , the attribute j and j' , and outputs the update key $UK_{j \rightarrow j'}$.

CS.KeyUpdate ($SK_{U_{id},2}, UK_{j \rightarrow j'}$) $\rightarrow SK'_{U_{id},2}$. The secret key update algorithm inputs user's secret key $SK_{U_{id},2}$ and the update key $UK_{j \rightarrow j'}$, and outputs a new secret key $SK'_{U_{id},2}$.

Security model

(1) Selective security model for our scheme. Initialization. The adversary $\mathcal{A}$ submits a challenged access structure $\mathbb{A}^*$ to the challenger $\mathcal{C}$.

Setup. The challenger $\mathcal{C}$ runs the setup algorithm and sends the public parameters PP to the adversary $\mathcal{A}$ and keeps the master key MSK to itself.

Phase 1. The adversary $\mathcal{A}$ adaptively issues repeated secret keys corresponding to attribute sets $S_1, S_2, \dots, S_q$, where none of these attribute sets satisfy the access structure $\mathbb{A}^*$.

Challenge. The adversary $\mathcal{A}$ submits two equal-length messages M_0 and M_1 to $\mathcal{C}$. The challenger $\mathcal{C}$ randomly selects a bit $b \in \{0,1\}$ and encrypts the message M_b for the access structure $\mathbb{A}^*$. The challenger $\mathcal{C}$ sends the ciphertext CT to the adversary $\mathcal{A}$.

Phase 2. Phase1 is repeated.

Guess. The adversary $\mathcal{A}$ outputs a guess b' of b . If $b' = b$, the adversary $\mathcal{A}$ wins this game. The advantage of the adversary $\mathcal{A}$ in this game is defined as $Adv = |\Pr(b' = b) - \frac{1}{2}|$.

Definition 1. The proposed scheme is selective security if all polynomial time adversaries have at most a negligible advantage in the above game.

(2) IND-CKA security model. Setup. Repeat the above security model's setup.

Phase1. The adversary $\mathcal{A}$ adaptively issues polynomial following queries.

H_1, H_2 -Query. The adversary $\mathcal{A}$ can query the random oracle H_1 or H_2 .

Trapdoor Queries. The adversary $\mathcal{A}$ can query any keywords trapdoor.

Challenge. The adversary $\mathcal{A}$ submits two keywords w_0 and w_1 to the challenger $\mathcal{C}$, with the restriction that the adversary $\mathcal{A}$ has not queried the trapdoors of keywords w_0 and w_1 . The challenger $\mathcal{C}$ randomly chooses a bit $b \in \{0,1\}$ and generates the index I_b of keyword w_b .

Phase 2. Phase1 is repeated.

Guess. The adversary $\mathcal{A}$ outputs a guess b' of b . If $b' = b$, the adversary $\mathcal{A}$ wins this game. The advantage of the adversary $\mathcal{A}$ in this game is defined as $Adv = |\Pr(b' = b) - \frac{1}{2}|$.

Definition 2. The proposed scheme is IND-CKA secure if all polynomial time adversaries have at most a negligible advantage in the above game.

Concrete construction

In this section, we present a construction for a keyword searchable attribute-based encryption scheme with attribute update for cloud storage.

Phase 1: System initialization

The AA first defines an attribute universe as $L = \{1, 2, \dots, m\}$ and chooses three hash functions $H : \{0, 1\}^* \rightarrow \mathbb{G}_0$, $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_0$ and $H_2 : \mathbb{G}_1 \rightarrow \{0, 1\}^{\log p}$, which can be modeled as random

oracles. Then the CS creates a user identity list $SL = (id, S_{U_{id}}, SK_{U_{id,2}}, K_{U_{id}})$ and a file list $FL = (F, CT, I_W, E_k(F))$, which are initially empty. Finally, the AA executes the setup algorithm.

AA.Setup $(\lambda, L) \rightarrow (PP, MSK, PK_s, SK_s)$. The setup algorithm first chooses two multiplicative cyclic groups $\mathbb{G}_0$ and $\mathbb{G}_1$ of prime order p , a generator $g \in \mathbb{G}_0$ and a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. It then chooses $x_s \in \mathbb{Z}_p$ and lets $SK_s = x_s$ as the CS's secret key. And it computes $PK_s = g^{x_s}$ as the CS's public key and publishes it. And it also randomly chooses three elements $\alpha, \gamma, \mu \in \mathbb{Z}_p$. In addition, it chooses a random number $v_j \in \mathbb{Z}_p$ for each attribute $j \in L$. Finally, it outputs the public parameters PP and the master secret key MSK as follows:

$$PP = \{\mathbb{G}_0, \mathbb{G}_1, g, g^\gamma, g^\mu, e(g, g)^\alpha, \{PK_j = H(j)^{v_j}, PK'_j = g^{v_j} | j \in L\}\}$$

$$MSK = \{\gamma, \mu, g^\alpha, \{v_j | j \in L\}\}$$

Phase 2: Key generation

The AA first distributes an attribute set $S_{U_{id}} \subseteq L$ associates with user's identity id , when a user with identity id requests a registration in the system. Secondly, the AA randomly chooses a number $X_{U_{id}} \in \mathbb{Z}_p$ for the user and calculates $K_{U_{id}} = g^{X_{U_{id}}}$. Then, the AA executes key generation algorithm.

AA.KeyGen $(MSK, id, S_{U_{id}}) \rightarrow (SK_{U_{id,1}}, SK_{U_{id,2}}, X_{U_{id}}, A_{priv}, B_{pub})$. The key generation algorithm first randomly chooses $t_{id} \in \mathbb{Z}_p$, which t_{id} is a unique assigned to the user with identity id . For each attribute $j \in S_{U_{id}}$, it randomly chooses $v_j \in \mathbb{Z}_p$. Finally, it outputs the user's secret key:

$$SK_{U_{id,1}} = D = g^\alpha \cdot g^{t_{id}}$$

$$SK_{U_{id,2}} = \left\{ \forall j \in S_{U_{id}} : D_j = H(j)^{\frac{t_{id}}{v_j}}, D'_j = g^{\frac{t_{id}}{v_j}}, v_j \in MSK \right\}$$

Lets $A_{priv} = \mu$ as the user's search secret key and $B_{pub} = g^\mu$ as the user's search public key.

Finally, the AA sends $(X_{U_{id}}, SK_{U_{id,1}}, A_{priv})$ to the user and publishes the user's searches public key B_{pub} . And, the AA sends $(id, S_{U_{id}}, SK_{U_{id,2}}, K_{U_{id}})$ to the CS, the CS stores $(id, S_{U_{id}}, SK_{U_{id,2}}, K_{U_{id}})$ in the SL-list.

Phase 3: File encryption and keyword index generation

Step 1: The DO encrypts the files.

To get ciphertext $E_k(F)$, the DO encrypts file F with a symmetric key k by the symmetric encryption algorithm. Then DO encrypts the symmetric key k by the following encryption algorithm.

DO.Encrypt $(PP, k, (M, \rho)) \rightarrow CT$. Let $\mathbf{M}$ be an $l \times n$ matrix, and $\mathbf{M}_i$ be the vector corresponding to the i th row of matrix $\mathbf{M}$. The function ρ associates rows of matrix $\mathbf{M}$ to attributes. The encryption algorithm first chooses a random vector $\mathbf{v} = (s, y_2, \dots, y_n) \in \mathbb{Z}_p^n$. These elements of vector $\mathbf{v}$ will be used to share the random encryption exponent s . For $i = 1$ to l , it calculates $\lambda_i = \mathbf{M}_i \mathbf{v}^T$. It then randomly chooses numbers $r_1, \dots, r_l \in \mathbb{Z}_p$ and outputs the ciphertext CT :

$$CT = \{(\mathbf{M}, \rho), \tilde{C} = k \cdot e(g, g)^{\alpha s}, C = g^s,$$

$$C_i = (g^{v_{\rho(i)}})^{\lambda_i} H(\rho(i))^{v_{\rho(i)} r_i}, C'_i = (g^{v_{\rho(i)}})^{r_i}, i = 1, 2, \dots, l\}$$

Where $v_{\rho(i)}$ refers to the master key is associated with attribute $\rho(i) \in L$.

Step2: Index generation.

The DO extracts a keywords set $W = (w_1, w_2, \dots, w_{n'})$ from the file F . Then DO executes the following index generation algorithm.

DO.Index $(W, B_{pub}) \rightarrow I_W$. The index generation algorithm randomly chooses $\xi_i \in \mathbb{Z}_p$ for each keyword $w_i \in W$ and calculates $\tau_i = e((B_{pub})^{\xi_i}, H_1(w_i))$. It then lets $I_{w_i} = [I_1, I_2] = [g^{\xi_i}, H_2(\tau_i)]$. Finally, It outputs the keywords index set $I_W = \{I_{w_i}\}_{i \in \{1, 2, \dots, n'\}}$.

Finally, the DO sends the file $(CT, I_W, E_k(F))$ to the CS. When the CS receives the uploaded file $(CT, I_W, E_k(F))$, it picks a unique identifier F^* for the file $(CT, I_W, E_k(F))$. The CS stores $(F^*, CT, I_W, E_k(F))$ in the FL-list.

Phase 4: Trapdoor generation

Step 1: Authentication information generation.

U.AuthorizationKey $(PP, PK_s, X_{U_{id}}) \rightarrow K'_{U_{id}}$. To generate the authentication information, the authentication information generation algorithm chooses a random number $\chi \in \mathbb{Z}_p$ and calculates $K_1 = (PK_s)^\chi (g^\chi)^{\frac{1}{X_{U_{id}}}}$, $K_2 = g^\chi$. It outputs the authentication information $K'_{U_{id}}$:

$$K'_{U_{id}} = \{K_1, K_2\} = \left\{ (PK_s)^\chi (g^\chi)^{\frac{1}{X_{U_{id}}}}, g^\chi \right\}$$

Step 2: Trapdoor generation.

U.Trapdoor $(w, PK_s, A_{priv}) \rightarrow T_w$. The trapdoor generation algorithm randomly chooses a number $\eta \in \mathbb{Z}_p$ and calculates $T_1 = g^\eta$, $T_2 = (PK_s)^\eta \cdot (H_1(w))^{A_{priv}}$. It outputs the trapdoor T_w :

$$T_w = \{T_1, T_2\} = \{g^\eta, (PK_s)^\eta \cdot (H_1(w))^{A_{priv}}\} = \{g^\eta, (g^{x_s})^\eta (H_1(w))^\mu\}$$

Finally, the user sends his id , the authentication information $K'_{U_{id}}$ and the trapdoor T_w to CS.

Phase 5: Verification

CS.Verifying $(id, K'_{U_{id}}) \rightarrow (0, 1)$. The validation algorithm inputs user's identity id and the authentication information $K'_{U_{id}}$. The cloud server uses its own secret key $SK_s = x_s$ to calculate

$$K''_{U_{id}} = \frac{K_1}{(K_2)^{x_s}} = g^{\frac{\gamma}{X_{U_{id}}}}$$

and judges the equation $K_{U_{id}} = K''_{U_{id}}$ holds or not. If the equation holds, which means the user is a legitimate user, it outputs 1. Otherwise, it outputs 0 and the algorithm is terminated.

Phase 6: File retrieval

CS.Test $(I_W, T_w) \rightarrow (0, 1)$. The test algorithm inputs the keywords index set I_W and the user's search trapdoor T_w . The cloud server uses its own secret key $SK_s = x_s$ and user's trapdoor $T_w = \{T_1, T_2\}$ to calculate

$$\varphi = \frac{T_2}{(T_1)^{x_s}} = \frac{(PK_s)^\eta \cdot (H_1(w))^{A_{priv}}}{(g^\eta)^{x_s}} = \frac{(g^{x_s})^\eta \cdot H_1^\mu(w)}{(g^\eta)^{x_s}} = H_1^\mu(w)$$

It then accords to keywords index $I_{w_i} = [I_1, I_2] = [g^{\xi_i}, H_2(\tau_i)]$ to calculate

$$\varphi_1 = e(I_1, \varphi) = e(g^{\xi_i}, H_1^\mu(w)) = e((g^\mu)^{\xi_i}, H_1(w))$$

and judges the equation $H_2(\varphi_1) = I_2$ holds or not. If the equation holds, which means the test is successful it outputs 1. Otherwise, it outputs 0 and the algorithm is terminated.

Phase 7: Data decryption

Step 1: Partial decryption by CS.

The CS first obtains ciphertext CT corresponding to keywords index I_W in the FL-list and finds user's secret key $SK_{U_{id,2}}$ in the SL-list. If user's secret key $SK_{U_{id,2}}$ is not the SL-list, the algorithm ends. Otherwise, it executes pre-decryption algorithm.

CS.PreDecrypt ($CT, SK_{U_{id,2}}$) $\rightarrow CT'$. The pre-decryption algorithm inputs user's secret key $SK_{U_{id,2}}$ for an attribute set $S_{U_{id}}$ and a ciphertexts CT for access structure (M, ρ) . At present, we assume that the attribute set $S_{U_{id}}$ satisfies the access structure (M, ρ) and let I be defined as $I = \{i | \rho(i) \in S_{U_{id}}\}$. Then, let $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ be as set of constants such that, if $\{\lambda_i\}_{i \in I}$ are valid shares of the secret s according to M , then $\sum_{i \in I} \omega_i \lambda_i = s$. The pre-decryption algorithm calculates

$$\begin{aligned} A &= \frac{\prod_{i \in I} e(C_i, D_{\rho(i)})^{\omega_i}}{\prod_{i \in I} e(C'_i, D_{\rho(i)})^{\omega_i}} \\ &= \frac{\prod_{i \in I} e((g^{v_{\rho(i)}})^{\lambda_i} H(\rho(i))^{v_{\rho(i)} r_i}, g^{v_{\rho(i)}})^{\omega_i}}{\prod_{i \in I} e((g^{v_{\rho(i)}})^{r_i}, H(\rho(i))^{v_{\rho(i)}})^{\omega_i}} \\ &= \frac{\prod_{i \in I} e((g^{v_{\rho(i)}})^{\lambda_i}, g^{v_{\rho(i)}})^{\omega_i} \cdot e(H(\rho(i))^{v_{\rho(i)} r_i}, g^{v_{\rho(i)}})^{\omega_i}}{\prod_{i \in I} e((g^{v_{\rho(i)}})^{r_i}, H(\rho(i))^{v_{\rho(i)}})^{\omega_i}} \\ &= \frac{\prod_{i \in I} e(g^{\lambda_i}, g^{t_{id}})^{\omega_i} \prod_{i \in I} e(H(\rho(i))^{r_i}, g^{t_{id}})^{\omega_i}}{\prod_{i \in I} e(g^{r_i}, H(\rho(i))^{t_{id}})^{\omega_i}} \\ &= e(g, g^{t_{id}})^{\sum_{i \in I} \omega_i \lambda_i} \\ &= e(g, g)^{t_{id} s} \end{aligned}$$

Finally, The CS sends part-ciphertext $CT' = (\tilde{C}, C, A)$ and the encrypted file $E_k(F)$ to the user.

Step2: User decryption

U.PostDecrypt ($CT', SK_{U_{id,1}}$) $\rightarrow k$. The post-decryption algorithm inputs the partial decrypted ciphertext CT' and the user's secret key $SK_{U_{id,1}}$. The user executes post-decryption algorithm to calculate symmetric k as follows:

$$k = \frac{\tilde{C} \cdot CT'}{e(C, D)} = \frac{k \cdot e(g, g)^{zs} \cdot e(g, g)^{t_{id} s}}{e(g^s, g^{\alpha} \cdot g^{t_{id}})} = \frac{k \cdot e(g, g)^{zs} \cdot e(g, g)^{t_{id} s}}{e(g, g)^{zs} \cdot e(g, g)^{t_{id} s}}$$

Finally, the user gets the plaintext $F = D_k(E_k(F))$ by the symmetric key k .

Phase 8: Attribute update

Step1: Update key generation.

AA.UKeyGen (PP, MSK, j, j') $\rightarrow UK_{j \rightarrow j'}$. The update key generation algorithm inputs the public parameter PP , the master secret key MSK , the attribute j and j' . For attribute j' , the AA

finds the random number $v_j \in \mathbb{Z}_p$ in the master secret key MSK , and then the AA outputs the update key $UK_{j \rightarrow j'}$:

$$UK_{j \rightarrow j'} = \left\{ UK'_{j \rightarrow j'} = H(j)^{\frac{-t_{id}}{v_j}} \cdot H(j')^{\frac{t_{id}}{v_{j'}}}, UK''_{j \rightarrow j'} = \frac{v_j}{v_{j'}} \right\}$$

Finally, it sends user's identity id and update key $UK_{j \rightarrow j'}$ to CS.

Step2: The CS executes a secret key update.

CS.KeyUpdate ($SK_{U_{id,2}}, UK_{j \rightarrow j'}$) $\rightarrow SK'_{U_{id,2}}$. The secret key update algorithm inputs user's secret key $SK_{U_{id,2}}$ and the update key $UK_{j \rightarrow j'}$. The CS executes the secret key update algorithm, and outputs a new secret key $SK'_{U_{id,2}}$.

$$SK'_{U_{id,2}} = \{ \forall i \in S_{U_{id}} \{j\} : D_i = H(i)^{\frac{t_{id}}{v_i}}, D'_i = g^{\frac{t_{id}}{v_i}} \}$$

$$D_{j'} = H(j)^{\frac{t_{id}}{v_j}} \cdot UK'_{j \rightarrow j'}, D'_{j'} = g^{\frac{t_{id}}{v_j}} \cdot UK''_{j \rightarrow j'}$$

Security analysis

Selective security proof for our scheme

Theorem 1. Let $\psi_0, \psi_1, \mathbb{G}_0$ and $\mathbb{G}_1$ be defined as in the generic bilinear group model. For any adversary $\mathcal{A}$ that makes a total of at most q queries to the oracles for computing the group operations in $\mathbb{G}_0$ and $\mathbb{G}_1$, the bilinear map e and the interaction with the IND-sCP-CPA security game, then the advantage of the adversary $\mathcal{A}$ in the IND-sCP-CPA security game is $\mathcal{O}\left(\frac{q^2}{p}\right)$.

Proof. In the IND-sCP-CPA security game, the challenge ciphertext has part-ciphertext $\tilde{C}$ may be $k_0 e(g, g)^{\alpha s}$ or $k_1 e(g, g)^{\alpha s}$. As in the [2], we modify ciphertext in the IND-sCP-CPA security game, now assuming the challenge ciphertext $\tilde{C}$ which may be $e(g, g)^{\alpha s}$ or $e(g, g)^\theta$, where $\theta \in \mathbb{Z}_p$ is randomly selected and the adversary $\mathcal{A}$ needs to determine which is the case. Obviously, any adversary $\mathcal{A}$ has advantage ε in the IND-sCP-CPA security game may be converted into $\mathcal{A}$ has at least $\frac{\varepsilon}{2}$ advantage in the modified IND-sCP-CPA security game (there are two situations can be considered: one in which the adversary $\mathcal{A}$ must distinguish between $k_0 e(g, g)^{\alpha s}$ and $e(g, g)^\theta$; another in which the adversary $\mathcal{A}$ must distinguish between $k_1 e(g, g)^{\alpha s}$ and $e(g, g)^\theta$. Obviously, both of these are equivalent to the above modified IND-sCP-CPA security game).

Initialization. The adversary $\mathcal{A}$ first submits an access structure (M^*, ρ^*) to the simulator S . In order to simulate the modified IND-sCP-CPA game, and then we introduce some mathematical symbols in the general bilinear group model, and let $\psi_0(0) = g, \psi_1(1) = e(g, g)$ (we will write $\psi_0(x) = g^x, \psi_1(y) = e(g, g)^y$).

Setup. The simulator S randomly chooses $\alpha, \gamma, \mu \in \mathbb{Z}_p$, and calculates $g^\mu, g^\gamma, e(g, g)^\alpha$. When the adversary $\mathcal{A}$ queries hash value of H on any attribute j , if it did not be queried, the simulator S randomly chooses $x_j \in \mathbb{Z}_p$, and then calculates $H(j) = g^{x_j}$, and writes the results into the Hash list. Otherwise, it looks for the Hash list. For any attribute $j \in L$, the simulator S randomly chooses a number $v_j \in \mathbb{Z}_p$. It sets the public parameter PP and the master secret key MSK as:

$$PP = \{g^\gamma, g^\mu, e(g, g)^\alpha, \{PK_j = H(j)^{v_j} = g^{x_j v_j}, PK'_j = g^{v_j} | j \in L\}\}$$

$$MSK = \{\gamma, \mu, g^\alpha, \{v_j | j \in L\}\}$$

The simulator S sends public parameter PP to the adversary $\mathcal{A}$.

Phase 1. The simulator S answers secret key queries as following:

Secret key query. When $\mathcal{A}$ makes its m 'th key generation query for the attribute set S_m , with a constraint that attribute set S_m does not satisfy access structure $(\mathbf{M}^*, \rho^*)$. The simulator S randomly chooses $t_m \in \mathbb{Z}_p$, and then calculates $D = g^x \cdot g^{t_m}$. For any attribute $j \in S_m$, the simulator S randomly chooses $v_j \in \mathbb{Z}_p$ and calculates $D_j = H(j)^{\frac{t_m}{v_j}}, D'_j = g^{\frac{t_m}{v_j}}$. It outputs secret key:

$$SK = \left\{ D = g^x \cdot g^{t_m}, \forall j \in S_{U_m} : D_j = H(j)^{\frac{t_m}{v_j}} = (g^{x_j})^{\frac{t_m}{v_j}}, D'_j = g^{\frac{t_m}{v_j}} \right\}$$

Then, the simulator S sends SK to adversary $\mathcal{A}$.

Challenge. The adversary $\mathcal{A}$ submits two equal messages k_0 and k_1 to the simulator S . First, the simulator S executes encryption algorithm according to the access structure $(\mathbf{M}^*, \rho^*)$. Where $\mathbf{M}^*$ is an $l \times n$ matrix. The $\mathbf{M}_i^*$ is i th row of matrix $\mathbf{M}^*$. The function ρ^* which associates rows of matrix $\mathbf{M}^*$ to attributes. Secondly, the simulator S chooses a random vector $\mathbf{v} = (s, y_2, \dots, y_n) \in \mathbb{Z}_p^n$. These elements of vector $\mathbf{v}$ will be used to share the encryption exponent s . Where $\lambda_i = \mathbf{M}_i^* \mathbf{v}^T$ is constrained by the LSSS scheme. Then, the simulator S chooses a random variable $b \in \{0, 1\}$ and l random variable values $r_1, \dots, r_l \in \mathbb{Z}_p$ to get the encryption of $k_b \in \mathbb{G}_1$ as: $C = g^s, \tilde{C} = k_b e(g, g)^{zs}, C_i = g^{v_{\rho^*(i)} \lambda_i + x_{\rho^*(i)} v_{\rho^*(i)} r_i}$ and $C'_i = g^{v_{\rho^*(i)} r_i}$.

The ciphertext is

$$CT^* = \{(\mathbf{M}^*, \rho^*), C = g^s, \tilde{C} = k_b e(g, g)^{zs},$$

$$C_i = g^{v_{\rho^*(i)} \lambda_i + x_{\rho^*(i)} v_{\rho^*(i)} r_i}, C'_i = g^{v_{\rho^*(i)} r_i}, i = 1, 2, \dots, l\}$$

Finally, the simulator S sends the ciphertext CT^* to adversary $\mathcal{A}$.

Phase 2. Phase1 is repeated.

The adversary $\mathcal{A}$ terminates and returns a guess b' of b after many queries. At this point, the simulator S randomly chooses a value $\theta \in \mathbb{Z}_p$ to get the simulated challenge ciphertext via substituting $\tilde{C} = e(g, g)^\theta$ for $\tilde{C} = k_b e(g, g)^{zs}$. After the simulation, the simulator S returns the simulated challenge ciphertext to adversary $\mathcal{A}$.

Next, we analyze the simulator S simulation. We think that the simulator S simulation is flawless with a constraint "unexpected collision" does not occur in the querying of $\psi_0(x) = g^x$, $\psi_1(y) = e(g, g)^y$ for group operation $\mathbb{G}_0$ and $\mathbb{G}_1$. Thus, an "unexpected collision" occurs when two queries corresponding to two different rational functions v and v' , it causes that $v' - v = 0$ for some variables. (Where an oracle query is regard as a rational function $v = \frac{a}{c}$ [2]). Then, we make the following analysis of "unexpected collision":

Before substitution. By the Schwartz-Zippel lemma[35,36], the probability of the "unexpected collision" occurs in $\mathbb{G}_0$ and $\mathbb{G}_1$ at most is $\mathcal{O}\left(\frac{q^2}{p}\right)$.

After substitution. We consider what the adversary's view would have been if we set $\theta = \alpha s$. We will show that subject to the conditioning above, the the adversary's view would have been identically distributed. Since we are in the generic group model where each group element's representation is uniformly and independently chosen[2], the only way that the adversary's view can differ in the case of $\theta = \alpha s$ is if there are two queries v and v' into $\mathbb{G}_1$ is $v \neq v'$ but $v|_{\theta = \alpha s} = v'|_{\theta = \alpha s}$. We prove show that this does never happens.

Case. To structure $\gamma' \alpha s$, we know that θ only exists as $e(g, g)^\theta$ in this form. According to the simulation, the simulator S wants v and v' is related to the θ is by having some additive terms of the form $\gamma' \theta$. Therefore, we must have $v - v' = \gamma' \alpha s - \gamma' \theta$ for some constant $\gamma' \neq 0$. Then, we artificially add the query $v - v' + \gamma' \theta = \gamma' \alpha s$ to the adversary's queries. According to the

conditions which we have set, we prove that adversary $\mathcal{A}$ cannot construct the query for $e(g, g)^{\gamma' \alpha s}$. Otherwise, a collision occurs and the theorem proves.

In order to gain a better understand of the above situation. We analyze based on the information given to the adversary $\mathcal{A}$ by the simulation. In Table 2, we enumerate the possibility queries of all rational function in $\mathbb{G}_1$ by the adversary $\mathcal{A}$. Except those in which every monomial involves the variable μ , since the variable μ is not relevant to constructing term αs . Where the variables j and j' represents the attribute string, and m indicates secret key queries made by the adversary $\mathcal{A}$.

According to Table 2, the adversary $\mathcal{A}$ can construct a polynomial $\alpha s + t_m s$ is by pairing s with $\alpha + t_m$. In this way, the adversary $\mathcal{A}$ also constructs a query term containing $\gamma' \alpha s + \sum_{m \in T} \gamma'_m t_m s$ for some collections T and constant $\gamma'_m \neq 0$. But the goal of the adversary $\mathcal{A}$ is to obtain a query polynomial $\gamma' \alpha s$, so the adversary $\mathcal{A}$ must add the negative terms $\sum_{m \in T} \gamma'_m t_m s$ to cancel the terms $\sum_{m \in T} \gamma'_m t_m s$. To construct the negative terms $\sum_{m \in T} \gamma'_m t_m s$, the adversary $\mathcal{A}$ first constructs a query polynomial of the from $t_m s$ by pairing $v_{\rho^*(i)} \lambda_i + x_j v_{\rho^*(i)} r_i$ with $\frac{t_m}{v_j}$ with a constraint $\rho^*(i) = j$, as we know s is linear combinations of λ_i . For the other collections T'_m and constant $\gamma'_{(i,m,j)} \neq 0$, the adversary $\mathcal{A}$ can also construct a query polynomial as:

$$\begin{aligned} & \gamma' \alpha s + \sum_{m \in T} (\gamma'_m t_m s + \sum_{m \in T'} \gamma'_{(i,m,j)} (v_{\rho^*(i)} \lambda_i + x_j v_{\rho^*(i)} r_i) \cdot \frac{t_m}{v_j}) \\ &= \gamma' \alpha s + \sum_{m \in T} (\gamma'_m t_m s + \sum_{m \in T'} \gamma'_{(i,m,j)} (\lambda_i t_m + x_j v_{\rho^*(i)} r_i t_m)) + \dots \end{aligned}$$

Therefore, we do some analysis to give the conclusion of this proof:

1. The set of secret shares $L_m = \{\lambda_i : \rho^*(i) = j, (i, j) \in T'_m\}$ do not reconstruct secret s for some $m \in T$. Then term $t_m s$ will still be retained, and $\mathcal{A}$ cannot construct $\gamma' \alpha s$.
2. If for all $m \in T$ the set of secret shares $L_m = \{\lambda_i : \rho^*(i) = j, (i, j) \in T'_m\}$ allow reconstruction the secret s . In order to get $\gamma' \alpha s$, the adversary $\mathcal{A}$ may cancel the term $\gamma'_m t_m s$ by the combination of the terms $t_m \lambda_i$, but $\mathcal{A}$ dose not get the term $x_j v_{\rho^*(i)} t_m r_i$ by examining the Table 2, there is no term such that $\mathcal{A}$ can cancel this term $\gamma'_m t_m s$. Therefore, the adversary $\mathcal{A}$ cannot construct $\gamma' \alpha s$.

IND-CKA security proof

Theorem 2. Supposing that BDH assumption holds, our scheme is semantically secure against a chosen keyword attack in the random oracle model.

Table 2. Possible query types from the adversary.

$x_j x_{j'}$	$(v_{\rho^*(i)} \lambda_i + x_{\rho^*(i)} v_{\rho^*(i)} r_i) \cdot x_{j'}$	$x_j v_{\rho^*(i)} r_i$	$(x_j) \left(\frac{x_{j'} t_m}{v_{j'}} \right)$
$\frac{x_j t_m}{v_j}$	x_j	$\gamma' \alpha + t_m$	$\alpha s + t_m s$
$(v_{\rho^*(i)} \lambda_i + x_j v_{\rho^*(i)} r_i) \cdot (v_{\rho^*(i')} \lambda_{i'} + x_{j'} v_{\rho^*(i')} r_{i'})$	$(v_{\rho^*(i)} \lambda_{i'} + x_{j'} v_{\rho^*(i')} r_{i'}) \cdot (v_{\rho^*(i)} r_i)$	$(v_{\rho^*(i')} \lambda_{i'} + x_{\rho^*(i')} v_{\rho^*(i')} r_{i'}) \cdot \left(\frac{x_j t_m}{v_j} \right)$	$(v_{\rho^*(i)} \lambda_i + (x_{\rho^*(i)} v_{\rho^*(i)} r_i) \cdot \left(\frac{t_m}{v_j} \right)$
$v_{\rho^*(i)} \lambda_i + x_{\rho^*(i)} v_{\rho^*(i)} r_i$	$v_{\rho^*(i)} v_{\rho^*(i')} r_i r_{i'}$	$\left(\frac{x_{j'} t_m}{v_{j'}} \right) (v_{\rho^*(i)} r_i)$	$\left(v_{\rho^*(i)} r_i \right) \left(\frac{t_m}{v_j} \right)$
$v_{\rho^*(i)} r_i$	$\left(\frac{x_j t_m}{v_j} \right) \left(\frac{x_{j'} t_m}{v_{j'}} \right)$	$\left(\frac{x_j t_m}{v_j} \right) \left(\frac{t_m}{v_j} \right)$	$\frac{x_j t_m}{v_j}$
$\left(\frac{t_m}{v_j} \right) \left(\frac{t_m}{v_{j'}} \right)$	t_m / v_j	s	

<https://doi.org/10.1371/journal.pone.0197318.t002>

Proof. Suppose the adversary $\mathcal{A}$ is a malicious cloud server that has non-negligible advantage ε in breaking our constructed searchable encryption scheme. Suppose that the adversary $\mathcal{A}$ makes at most q_{H_2} hash function queries to H_2 and at most q_T trapdoor queries (we assume q_{H_2} and q_T are positive). We will construct a simulator $\mathcal{B}$ to solve BDH problem with advantage $\varepsilon' = 2\varepsilon/(eq_{H_2}q_T)$, where e is the base of the natural.

Initialization. The simulator $\mathcal{B}$ receives a BDH challenge and chooses two multiplicative cyclic groups $\mathbb{G}_0$ and $\mathbb{G}_1$ of prime order p , a generator $g \in \mathbb{G}_0$ and a bilinear map $e: \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. Then simulator $\mathcal{B}$ randomly chooses $a, b, c \in \mathbb{Z}_p$, lets $u_1 = g^a, u_2 = g^b, u_3 = g^c \in \mathbb{G}_0$, its aim is to compute $e(g, g)^{abc} \in \mathbb{G}_1$.

Setup. The simulator $\mathcal{B}$ randomly chooses a number $x_s \in \mathbb{Z}_p$, lets public key $PK_s = g^{x_s}$ and secret key $SK_s = x_s$ for the adversary $\mathcal{A}$. To simulate the user's search public keys B_{pub} and secret keys A_{priv} , the simulator $\mathcal{B}$ chooses a random parameter $t_1 \in \mathbb{Z}_p$ and sets $B_{pub} = u_1^{t_1}$, so $A_{priv} = \mu = at_1$.

Phase1. The adversary $\mathcal{A}$ adaptively issues following queries:

H_1 -Query: The adversary $\mathcal{A}$ can query the random oracle H_1 at any time. To answer to H_1 queries, the simulator $\mathcal{B}$ maintains a list of tuples (w_i, h_i, e_i, c_i) called the H_1 -list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_1 of any keywords $w_i \in \{0,1\}^*$, the simulator $\mathcal{B}$ answers as follows:

1. If the query w_i has already appeared on the H_1 -list in a tuple (w_i, h_i, e_i, c_i) , the simulator $\mathcal{B}$ responds with $H_1(w_i) = h_i \in \mathbb{G}_0$.
2. Otherwise, $\mathcal{B}$ generates a random coin $c_i \in \{0,1\}$ so that $\Pr[c_i = 0] = \frac{1}{q_T+1}$, where q_T is a trapdoor query.

If $c_i = 0$, the simulator $\mathcal{B}$ calculates $h_i = u_2^{e_i} \in \mathbb{G}_0$;

If $c_i = 1$, the simulator $\mathcal{B}$ calculates $h_i = g^{e_i} \in \mathbb{G}_0$, where the value $e_i \in \mathbb{Z}_p$ randomly is selected. Then $\mathcal{B}$ adds the tuple (w_i, h_i, e_i, c_i) to the H_1 -list, and returns $H_1(w_i) = h_i$ to the adversary $\mathcal{A}$.

H_2 -Query: $\mathcal{A}$ can query the random oracle H_2 at any time. To answer to H_2 queries, the simulator $\mathcal{B}$ maintains a list of tuples (t_i, V_i) called the H_2 -list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_2 of any $t_i \in \mathbb{G}_1$, the simulator $\mathcal{B}$ answers as follows:

1. If the query t_i has already appeared on the H_2 -list in a tuple (t_i, V_i) , the simulator $\mathcal{B}$ responds with $H_2(t_i) = V_i \in \{0,1\}^{logp}$.
2. Otherwise, the simulator $\mathcal{B}$ randomly chooses a value $V_i \in \{0,1\}^{logp}$, and the simulator $\mathcal{B}$ adds the tuple (t_i, V_i) to the H_2 -list, and returns $H_2(t_i) = V_i$ to $\mathcal{A}$.

Trapdoor queries: When $\mathcal{A}$ queries the trapdoor of any keywords $w_i \in \{0,1\}^*$, the simulator $\mathcal{B}$ first executes the H_1 queries to obtain $h_i \in \mathbb{G}_0$ such that $H_1(w_i) = h_i$ and (w_i, h_i, e_i, c_i) corresponding to the tuple on the H_1 -list. the simulator $\mathcal{B}$ answers as follows:

1. If $c_i = 0$, the simulator $\mathcal{B}$ declares the failure and aborts.
2. If $c_i = 1$, $h_i = g^{e_i} \in \mathbb{G}_0$. the simulator $\mathcal{B}$ randomly chooses a value $\eta \in \mathbb{Z}_p$, and calculates

$$T_1^* = g^\eta, T_2^* = (g^{x_s})^\eta \cdot (B_{pub})^{e_i} = (g^{x_s})^\eta (u_1^{t_1})^{e_i} = (PK_s)^\eta H_1(w_i)^\mu$$

The simulator $\mathcal{B}$ sends trapdoor $T^* = \{T_1^*, T_2^*\}$ to $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ submits a pair of keywords w_0 and w_1 , where keywords w_0 and w_1 trapdoor had not been queried by $\mathcal{A}$. the simulator $\mathcal{B}$ generates keyword index as follows:

1. the simulator $\mathcal{B}$ first executes H_1 queries twice to obtain $h_0, h_1 \in \mathbb{G}_0$ such that $H_1(w_0) = h_0$, $H_1(w_1) = h_1$. For $i = 0, 1$, we set (w_i, h_i, e_i, c_i) corresponding to the tuple on the H_1 -list. If both $c_0 = 0$ and $c_1 = 1$, then $\mathcal{B}$ declares the failure and aborts.
2. we known that at least one of c_0 and c_1 is equal to 0. The simulator $\mathcal{B}$ chooses a bit $b \in \{0, 1\}$ such that $c_b = 0$.
3. The simulator $\mathcal{B}$ answers the keyword index $I_W^* = (I_1^*, I_2^*)$. The simulator $\mathcal{B}$ then randomly chooses a parameter $t_2 \in \mathbb{Z}_p$ and sets $I_1^* = u_3^{1/t_2}$ with the implied setting $(\xi_i = c/t_2)$, where c is unknown, and we know $u_3 = g^c$ is a part of BDH.
4. the simulator $\mathcal{B}$ randomly chooses a $Z \in \{0, 1\}^{\log p}$, and sets $I_2^* = Z$.

With the definition, $I_W^* = (I_1^*, I_2^*)$ is an effective keyword index for keyword w_b as queried.

Phase 2. Phase1 is repeated.

Output. The adversary $\mathcal{A}$ outputs its guess b' of bit b .

Note the values $h_i = u_2^{e_i}$ be set with probability $\frac{1}{q_T+1}$ in the setting of the H_1 queries. Since $\mathcal{A}$ queries the value of the form to H_2 oracle

$$e((B_{pub})^{\xi_i}, H_1(w_b)) = e((g^{at_1})^{\xi/t_2}, g^{be_b}) = e(g, g)^{abc(e_b t_1/t_2)}$$

with the same probability $\frac{1}{q_T+1}$ in the setting of the H_2 -list, therefore

$$(e(g, g)^{abc(e_b t_1/t_2)}, H_2(e((B_{pub})^{\xi_i}, H_1(w_b)))) \in H_2$$

Then, the simulator $\mathcal{B}$ randomly chooses a pair $(t_i, V_i) \in H_2$ -list and outputs $t^{e_b t_1/t_2}$ as its guess for $e(g, g)^{abc}$. Where t_1, t_2 and e_b are set according to the parameters of the challenge phase.

Probability Analyses. We can prove that $\mathcal{A}$ can win the game with a non negligible probability, then $\mathcal{B}$ can solve the BDH problem with the probability at least $\frac{2\varepsilon}{eq_T q_{H_2}}$. The specific probability analysis is similar to the scheme[34].

Because of the BDH assumption that the BDH problem is tough, so the probability $\frac{2\varepsilon}{eq_T q_{H_2}}$ is negligible. So that our scheme is secure under the BDH assumption.

Computational complexity and performance evaluation

Computational complexity comparison

In Table 3, we give the comparison of the computational complexity of our scheme with the schemes [13, 19, 21, 31]. As shown in Table 3, our scheme has a less amount of computation in the

Table 3. Comparison of computational complexity.

schemes	[13]	[19]	[21]	[31]	ours
PP	$(m+4)e+p$	$(m+2)e+p$	$(m+2)e+p$	$(2m+2)e+p$	$(2m+3)e+p$
SK	$(2l_1+11)e$	$(3l_1+5)e$	$(l_1+5)e$	$(3l_1+2)e$	$(2l_1+2)e$
CT	$(3l_2+4)e+p$	$(3l_2+4)e+p$	$(l_2+4)e+p$	$(2l_2+2)e+p$	$(3l_2+2)e+p$
KeyUpdate	l_3e	$(2l_3+1)e$	l_3e	l_3e	$(l_3+1)e$
CTUpdate	l_4e	l_4e	l_4e	l_4e	X
Trapdoor	X	X	X	$e+p$	$2e$
Index	X	X	X	$e+2p$	$e+2p$

(Continued)

Table 3. (Continued)

schemes	[13]	[19]	[21]	[31]	ours
CS. Decrypt	$\mathbf{X}$	$\mathbf{X}$	$2e + (l_5 + 2)p$	$\mathbf{X}$	$(l_5 + 1)p$
User. Decrypt	$2e + (l_5 + 2)p$	$4e + (l_5 + 6)p$	$2e + p$	$2e + (1 + l_5)p$	$2e + p$

e : an exponential operation in $\mathbb{G}_0, \mathbb{G}_1$

p : a pairing operation

m : the number of attributes in universe

l_1 : the number of attributes in the user's private key

l_2 : the number of attributes in the access structure

l_3 : the number of user attributes that need to be updated

l_4 : the number of attributes in the ciphertext that need to be updated

l_5 : the number of user's attributes satisfying an access control

$\mathbf{X}$: there is no corresponding function or process in the scheme.

<https://doi.org/10.1371/journal.pone.0197318.t003>

key generation and encryption generation compared with the schemes in [13,19,31]. Actually, our scheme needs the minimum amount of computation when the users decrypt the ciphertext. And most important is that our scheme need not update ciphertext when an attribute update occurs, which also help us greatly reduce the amount of computation. In addition, our scheme have the

(a) Setup generation time

(b) Key generation time

(c) Encryption time

(d) Decryption time of the user

Fig 2. Performance evaluation. (a) Setup generation time (b) Key generation time (c) Encryption time (d) Decryption time of the user.

<https://doi.org/10.1371/journal.pone.0197318.g002>

function of the keyword search, which can make the search more efficiently and more accurately. The schemes of [13],[19] and [21] don't achieve the function of keyword search.

Performance evaluation

To evaluate the performance of our scheme and the scheme [19], we simulate the computational time of the setup generation, key generation, encryption and decryption by user with different number of attributes. As shown in Fig 2. The implementation is executed by using of the Pairing Cryptography (PBC) library[37]. We can clearly see from Fig 2(A) that the setup generation times scales linearly in the number of attribute in attribute universe in both scheme. Fig 2(B) shows secret key times scales linearly in the number of attribute in secret key in both scheme. Fig 2(C) shows the encryption times scales linearly in the number of attribute in ciphertexts in both scheme. The setup generation time is shown in Fig 2(A). We find also that the setup generation takes higher computational time in our scheme than the scheme [19]. The key generation time is shown in Fig 2(B) and the encryption time is shown in Fig 2(C). Obviously, the encryption time and key generation time of the scheme[19] takes higher computational time than our scheme. Fig 2(D) shows that the user- decryption time of our scheme takes lesser computational time than the scheme [19].

Conclusions

In this paper, we have proposed a keyword searchable attribute-based encryption scheme with attribute update for cloud storage. Our new scheme supports both the user's attribute update and supports multi-user keywords search, as long as user's trapdoor could match keyword index stored in the cloud storage, then the user can search interesting encrypted file successfully. The performance evaluation results confirm that the proposed scheme is more efficient than other attribute based encryption schemes with attribute update. In addition, we outsource the operation with high computation cost to the cloud storage to reduce the user's computational burden. Moreover, our scheme also is proven to be semantic security against chosen ciphertext-policy and chosen plaintext attack in the general bilinear group model.

Supporting information

S1 File.
(ZIP)

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants 61572019,61173192, the Key Project of Research Foundation of Natural Science Foundation of Shaanxi Province of China under Grant NO.2016JZ001,the Key Laboratory Research Project of Education Bureau of Shaanxi Province of China under Grant No.16JS078.Thanks also go to the anonymous reviewer for their useful comments.

Author Contributions

Formal analysis: Shangping Wang.

Methodology: Shangping Wang.

Writing – original draft: Jian Ye.

Writing – review & editing: Yaling Zhang.

References

1. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data[C]// ACM Conference on Computer and Communications Security. ACM, 2006:89–98.
2. Bethencourt J, Sahai A, Waters B. Ciphertext-Policy Attribute-Based Encryption[C]// IEEE Symposium on Security and Privacy. IEEE Computer Society, 2007:321–334.
3. Ostrovsky R, Sahai A, Waters B. Attribute-based encryption with non-monotonic access structures[C]// Ccs 07 Acm Conference on Computer & Communications Security. 2007:195–203.
4. Waters B. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization[J]. *Lecture Notes in Computer Science*, 2011, 2008:321–334.
5. Chase M, Chow S S M. Improving privacy and security in multi-authority attribute-based encryption[C]// ACM Conference on Computer and Communications Security. ACM, 2009:121–130.
6. Hur J. Improving Security and Efficiency in Attribute-Based Data Sharing[J]. *Knowledge & Data Engineering IEEE Transactions on*, 2013, 25(10):2271–2282.
7. Liu X, Ma J, Xiong J, Li Q, Ma J. Ciphertext-Policy Weighted Attribute Based Encryption for Fine-Grained Access Control[C]// International Conference on Intelligent NETWORKING and Collaborative Systems. IEEE, 2014:51–57.
8. Lai J, Deng R H, Li Y, Weng J. Fully secure key-policy attribute-based encryption with constant-size ciphertexts and fast decryption[C]// ACM Symposium on Information, Computer and Communications Security. ACM, 2014:239–248.
9. Horváth M. Attribute-Based Encryption Optimized for Cloud Computing[M]// SOFSEM 2015: Theory and Practice of Computer Science. Springer Berlin Heidelberg, 2015:1–9.
10. Attrapadung N, Imai H. Attribute-Based Encryption Supporting Direct/Indirect Revocation Modes[C]// Ima International Conference on Cryptography and Coding. Springer-Verlag, 2009:278–300.
11. Li Y, Zhu J, Wang X, Shao S. Optimized Ciphertext-Policy Attribute-Based Encryption with Efficient Revocation[J]. *International Journal of Security & Its Applications*, 2013, 7(6):385–394.
12. Zhang Y, Chen X, Li J, Li H, Li F. FDR-ABE: Attribute-Based Encryption with Flexible and Direct Revocation[C]// International Conference on Intelligent NETWORKING and Collaborative Systems. IEEE, 2013:38–45.
13. Wang H, Zheng Z, Wu L, Li P. New directly revocable attribute-based encryption scheme and its application in cloud storage environment [J]. *Cluster Computing*, 2016:1–8.
14. Tu S, Niu S, Li H. A fine-grained access control and revocation scheme on clouds[J]. *Concurrency & Computation Practice & Experience*, 2016, 28(6):1697–1714.
15. Yu S, Wang C, Ren K, Lou W. Attribute based data sharing with attribute ACM revocation [C]// Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April. DBLP, 2010:261–270.
16. Qian H, Li J, Zhang Y, Han J. Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation [J]. *International Journal of Information Security*, 2015, 14(6):487–497.
17. Huang X F, Tao Q, Qin B D, Liu Z Q. Multi-Authority Attribute Based Encryption Scheme with Revocation[C]// International Conference on Computer Communication and Networks. IEEE, 2015:1–5.
18. Li Q, Feng D, Zhang L. An attribute based encryption scheme with fine-grained attribute revocation[C]// Global Communications Conference. IEEE, 2012:885–89–890.
19. Zhang P, Chen Z, Liang K, Wang S, Wang T. A Cloud-Based Access Control Scheme with User Revocation and Attribute Update[C]// Asian Conference on. Springer-Verlag New York, Inc. 2016:525–540.
20. Liao J, Jiang C, Guo C. Data privacy protection based on sensitive attributes dynamic update[C]// International Conference on Cloud Computing and Intelligence Systems. IEEE, 2016:377–381.
21. Zhang P, Chen Z, Liu J K, Liang K, Liu H. An efficient access control scheme with outsourcing capability and attribute update for fog computing [J]. *Future Generation Computer Systems*, 2016. <https://doi.org/10.1016/j.future.2015.09.006> PMID: 26688598
22. Song D X, Wagner D, Perrig A. Practical Techniques for Searches on Encrypted Data[C]// Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on. IEEE, 2002:44.
23. Dan B, Crescenzo G D, Ostrovsky R, Persiano G. Public Key Encryption with Keyword Search[J]. *Lecture Notes in Computer Science*, 2003, 3027(16):506–522.
24. Dan B, Waters B. Conjunctive, Subcollection, and Range Queries on Encrypted Data[C]// The Theory of Cryptography Conference. 2006:535–554.
25. Cao N, Wang C, Li M, Lou W J. Privacy-preserving multi-keyword ranked search over encrypted cloud data[C]// INFOCOM, 2011 Proceedings IEEE. IEEE, 2011:829–837.

26. Zhen hua L, Jin miao W, Bo L. A ciphertext-policy hidden vector encryption scheme supporting multi-user keyword search[J]. *Security & Communication Networks*, 2015, 8(6):879–887.
27. Fu Z, Sun X, Liu Q, Shu J. Achieving Efficient Cloud Search Services: Multi-Keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing[J]. *leice Trans Commun*, 2015, 98(1):190–200.
28. Li H, Liu D, Jia K, Lin X. Achieving authorized and ranked multi-keyword search over encrypted cloud data[C]// *IEEE International Conference on Communications*. IEEE, 2015:7450–7455.
29. Lv Z, Zhang M, Feng D. Multi-user Searchable Encryption with Efficient Access Control for Cloud Storage[C]// *IEEE, International Conference on Cloud Computing Technology and Science*. IEEE, 2014:366–373.
30. 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[J]. *IEEE Transactions on Parallel & Distributed Systems*, 2016, 27(4):1187–1198. <https://doi.org/10.1016/j.jbiomech.2005.09.015> PMID: 16325826
31. Wang S, Zhang X, Zhang Y. Efficiently Multi-User Searchable Encryption Scheme with Attribute Revocation and Grant for Cloud Storage:[J]. *Plos One*, 2016, 11(11):e0167157. <https://doi.org/10.1371/journal.pone.0167157> PMID: 27898703
32. Wang S, Zhao D, Zhang Y. Searchable attribute-based encryption scheme with attribute revocation in cloud storage[J]. *Plos One*, 2017, 12(8):e0183459. <https://doi.org/10.1371/journal.pone.0183459> PMID: 28859125
33. Zu L, Liu Z, Li J. New Ciphertext-Policy Attribute-Based Encryption with Efficient Revocation[C]// *IEEE International Conference on Computer and Information Technology*. IEEE, 2014:281–287.
34. Rhee H S, Park J H, Susilo W, Dong H L. Trapdoor security in a searchable public-key encryption scheme with a designated tester ☆[J]. *Journal of Systems & Software*, 2010, 83(5):763–771.
35. Schwartz J. T. Fast Probabilistic Algorithms for Verification of Polynomial Identities[M]// *Symbolic and Algebraic Computation*. Springer Berlin Heidelberg, 1979:10–1145.
36. Zippel R. Probabilistic algorithms for sparse polynomials[C]// *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposium on Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings. DBLP*, 1979:216–226.
37. Duquesne S, Lange T. Pairing-based cryptography[J]. *Math.iisc.ernet.in*, 2005, 22(3):573–590.