PLOS ONE

RESEARCH ARTICLE

Efficiently Multi-User Searchable Encryption Scheme with Attribute Revocation and Grant for Cloud Storage

Shangping Wang[1], Xiaoxue Zhang[1]*, Yaling Zhang[2]

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

* iszhangxiaoxue@163.com

Citation: Wang S, Zhang X, Zhang Y (2016) Efficiently Multi-User Searchable Encryption Scheme with Attribute Revocation and Grant for Cloud Storage. PLoS ONE 11(11): e0167157. doi:10.1371/journal.pone.0167157

Editor: Houbing Song, West Virginia University, UNITED STATES

Received: July 30, 2016

Accepted: November 9, 2016

Published: November 29, 2016

Data Availability Statement: All relevant data are within the paper.

Funding: 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 Research Foundation of Education Department of Shaanxi Province of China under grants 2013JK1142, and the Research Foundation of science and technology of Xi'an Beilin district of China under grants GX1407.

Abstract

Cipher-policy attribute-based encryption (CP-ABE) focus on the problem of access control, and keyword-based searchable encryption scheme focus on the problem of finding the files that the user interested in the cloud storage quickly. To design a searchable and attribute-based encryption scheme is a new challenge. In this paper, we propose an efficiently multi-user searchable attribute-based encryption scheme with attribute revocation and grant for cloud storage. In the new scheme the attribute revocation and grant processes of users are delegated to proxy server. Our scheme supports multi attribute are revoked and granted simultaneously. Moreover, the keyword searchable function is achieved in our proposed scheme. The security of our proposed scheme is reduced to the bilinear Diffie-Hellman (BDH) assumption. Furthermore, the scheme is proven to be secure under the security model of indistinguishability against selective ciphertext-policy and chosen plaintext attack (IND-sCP-CPA). And our scheme is also of semantic security under indistinguishability against chosen keyword attack (IND-CKA) in the random oracle model.

I. Introduction

The fuzzy identity based encryption (IBE) which is regarded as the prototype of attribute-based cryptography was put forward by Sahai and Waters [1] in 2005. In an attribute-based encryption system, each user has a number of descriptive attributes (such as gender, age, education, occupation, etc.). Meanwhile, the users' private key and ciphertext are link with some described attribute set and access strategy respectively. When the private key is matched with ciphertext, the user can decrypt the ciphertext.

Goyal et al. [2] put the ABE scheme into CP-ABE scheme and the key-policy attribute-based encryption (KP-ABE) scheme, and definitions are given respectively.

Bethencourt et al. [3] provided a new structure. The scheme can not only achieve a flexible access structure but also has an important characteristic of anti-collusion. That is, different users can not add their own access right by collusion their private key. Besides, there are some other outstanding articles such as the scheme proposed by Emura et al. [4] which has a certain contribution to the computational complexity and storage load.

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

The above-mentioned CP-ABE schemes have made outstanding contributions, but due to the constant changes of the realistic situation, the schemes still face new challenges. Once some users' attributes change, the system should timely update these users' attribute set and the corresponding private key.

A number of programs research about attribute revocation have been put forward [5–13]. Generally speaking, the revocation mechanism can be divide into two types: direct revocation scheme [6–9] and indirect revocation scheme [10–13]. The big difference between them is that the direct revocation scheme is enforced by a specified revocation list and indirect revocation scheme is enforced by updating the private key of the non-revoked users (Implicitly, the revoked users' private key are revoked). Zhang et al. [8] put forward a scheme with direct revocation, which characterized by the fact that the length of the encrypted text is fixed, and the partial ciphertext update is only required when the revocation occurs. The scheme put forward by Yu et al. [10] achieves an efficient encryption update through proxy re encryption. But there is a limit to the scheme. That is the fixed strategy. Due to the high efficiency and the limitation of the scheme, Naruse et al. [13] made a further study of this article. The scheme proposed by them can be applied to a more flexible access strategy.

On the other hand, due to the continuous development of computer network and the outsourcing technology many enterprises began to establish their own local network and database. Through the establishment of a certain data encryption and an access control, they passed their database to a third party management. Since the third party is not credible, efficient search capability and secure search process are two important tasks in the present study. Some articles research on these two directions have been put forward [14–17].

Bao at al. [14] put forward a scheme which can be applied to the cloud storage environment. This program can realize the multi user search process. Because of the users' access rights in the system are different according to their own attribute set. The efficiency of system should be further improved with the increase of users' number.

Some schemes research on highly efficient access control of multi user keyword search have been put forward [18–24].

Recently, Lv et al. [18] proposed an efficient keyword searchable model. However, the scheme does not have a complete security model. When a user's attributes in the system change, the limitations of the program appeared. Kaci et al. [23] put forward a scheme that is consistent with ACAS (Access Control Aware Search) principle and improve the level of confidentiality of outsourced data. Nonetheless, the efficiency of the proposed model is evaluated according to data size.

Most of the existing multi user attribute-based keyword searchable encryption schemes focus on efficient access control and fast search process, of which there are some articles can achieve revocation of user, for example by removing user's search key in proxy server to achieve revocation [18].

In addition, some research on the security of information under specific scenarios are also proposed. Specifically, a first research direction focuses on the security of Vehicular and hoc network [25–27]. A second research direction deal with the security communication in Internet of Things (IOT) networks [28, 29]. There are other research directions, such as file search in unstructured P2P (peer-to-peer) gird networks [30–32] and WSNs (wireless sensor networks) in healthcare applications [33] etc.

A. Our Contributions

- Our scheme supports user's multiple attributes revocation and grant simultaneously by adding a series of attribute parameters. The attribute revocation in our scheme is a fine grain

Table 1. Comparisons of Our Scheme with the Main References.

scheme	access control	keyword searchable	attribute revocation for some user	attribute grant for some user	user revocation	lazy revocation
[3]	LSSS	✗	✗	✗	✗	✗
[18]	access tree	✓	✗	✗	✓	✗
[13]	LSSS	✗	✓	✓	✗	✗
Ours	access tree	✓	✓	✓	✓	✓

✓: The scheme has the function.

✗: The scheme does not have this function.

doi:10.1371/journal.pone.0167157.t001

method. That is, our revocation is able to revoke some users' some attributes, rather than to revoke a single attribute or revoke attributes in the system. The attribute grant method is similarly. In addition, the proposed scheme is proven to be IND-sCP-CPA secure.

- We use a lazy revocation technique [34] for user's attribute and private key update process. It is to say that only when user accesses the encrypted files, it helps to update the user's attribute and private key.
- As keyword searchable process in [18] does not have a complete security proof. By changing the operation of search trapdoor in [18], we have proved that our proposed keyword searchable scheme is IND-CKA secure in the random oracle model under bilinear Diffie-Hellman (BDH) assumption.
- The function of revocation of user identity in our scheme is consistent with that in [18].

B. Comparisons

We compare the function of our scheme with the existing schemes presented in [3, 13, 18] in Table 1.

II. Preliminaries

A. Mathematical Tools

We first give some of the mathematical tools will be used later in this article, the specific argument can be found in the references.

Definition 1 (Bilinear Map [2]). The definition of the two multiplication of group $\mathbb{G}_1$ and $\mathbb{G}_2$, so that their order is p and the generator of $\mathbb{G}_1$ is g. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, which satisfies:

- *Bilinearity*: for all $u,\ v \in \mathbb{G}_1$ and $a,\ b \in \mathbb{Z}_p,\ e(u^a, v^b) = e(u, v)^{ab}$;
- *Non-degeneracy*: $e(g, g) \neq 1$;
- *Computability*: for all $u,\ v \in \mathbb{G}_1$, $e(u, v)$ is efficiently computable.

Definition 2 (Lagrange Coefficient [24]). The definition of a Lagrange coefficient is $\Delta_{i,S}(x)$, which $i \in \mathbb{Z}_p$ and the elements of set S belong to $\mathbb{Z}_p$. Then we have the following equation:

$$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x - j}{i - j}$$

B. Access Tree

In this paper, we use the access tree as the access policy.

Definition 3 (Access Tree [18]). In the access tree, the number of child nodes of each interior node x is denoted as num_x. The threshold value of each node is defined as (k_x, num_x), which is $0 < k_x \leq num_x$. In particular, when $k_x = 1$ threshold for an ′OR′ gate. When $k_x = num_x$ for an ′AND′ gate. Furthermore, each leaf node are correlated and attribute. For the convenience of using access tree, we define several functions as follow.

- $parent(x)$: this returns the parent node of a node x except the root node.
- $index(x)$: assuming that the children nodes of each node are numbered from 1 to num, this returns such a number associated with the node x.
- $att(x)$: this returns the attribute associated with a leaf node x.

C. BDH Problem

Choose two cyclic group $\mathbb{G}_1$ and $\mathbb{G}_2$, enable their order is p. And a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a valid bilinear map. BDH problem under the tuple $< g,\ \mathbb{G}_1,\ \mathbb{G}_2,\ e >$ can be defined as: fix a generator g of $\mathbb{G}_1$, as well as g^a, g^b, g^c for some random $a,\ b,\ c \in \mathbb{Z}_p$, compute $e(g,g)^{abc} \in \mathbb{G}_2$.

BDH assumption [35]. The assumption is valid if there is no polynomial-time adversary can be non-negligible probability to solve the above BDH problem.

III. System and Security Models

A. System Model

First, define five entities of the system: an attribute authority, dada owners, a proxy server, a cloud sever, users, can be described below. The system model of our scheme is given in Fig 1.

- **Attribute authority (AA).** Attribute authority is entirely credible to other entities and is responsible for the system establishment, new user register, attributes assignment and key generation. When some users' attribute set change (that is, some attributes are revoked or granted), AA establishes a revoked and a granted user list set for each attribute respectively and updates the public parameter, master key, proxy update key and proxy grant key.
- **Data owner (DO).** Data owner is responsible for uploading all the data files to cloud server. In order to ensure other legitimate users of the system can search for the corresponding file through the keyword, data owner needs to extract keywords and establish keyword indexes. Finally, along with the encrypted files upload to cloud server.
- **User (U).** Legitimate users can download their interest files from the system. In order to hide the search keyword, the user generates a search trapdoor. And then sends his unique identity, attribute set, partial private key component to the proxy server for updating attribute set and private key component. After receiving the updated attribute set and private key component, he sends his trapdoor together with his unique identity to cloud server. Without revealing any information about the content of the file, proxy server to help complete most of the decryption work. And then the final message is calculated by the user.
- **Proxy server (PS).** Proxy server is deployed by AA. It re-encrypts encrypted shared data and updats user's attribute set and corresponding private key by using the proxy update key and proxy grant key received from AA. It also can help the users execute most CP-ABE decryption task.

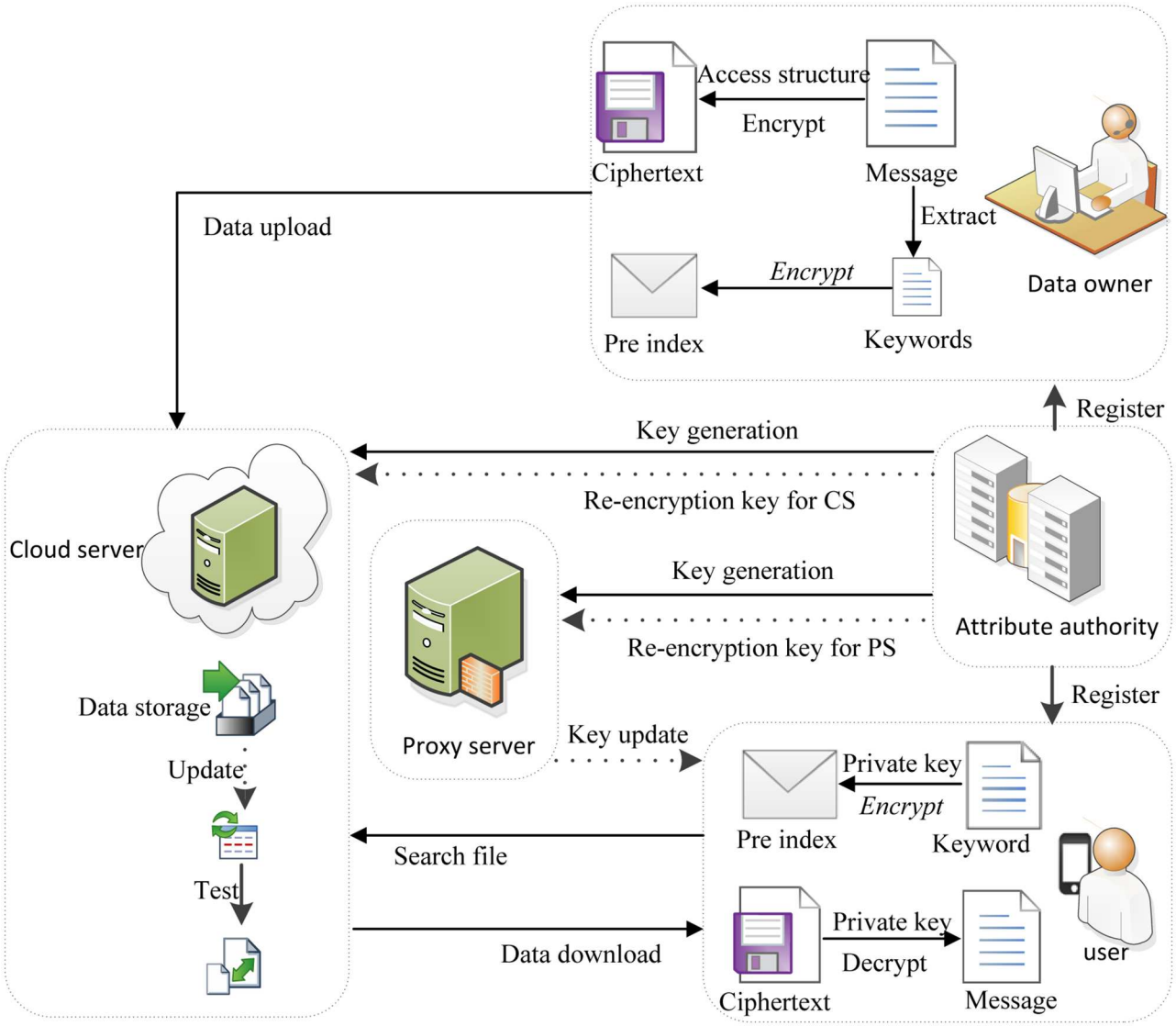

Fig 1. System model of the proposed scheme.

doi:10.1371/journal.pone.0167157.g001

- **Cloud server (CS).** This paper mainly use the large storage characteristics of CS to store the data files in the system. Besides, it also helps to generate keyword index and trapdoor. In order to achieve efficient search we use the *D. Data Upload* method in [18] to store and search files. Also similar to the *G. User Revocation* method in [18], CS can perform user revocation operation.

B. Algorithms Definitions

Our proposed efficiently multi-user searchable encryption scheme with attribute revocation and grant for cloud storage is composed of thirteen randomized polynomial time algorithms.

- **AA.Setup** $(\lambda, U) \rightarrow (PP, MK, UK, GK)$: The setup algorithm takes a security parameter λ and an attribute universe description U as input. It outputs the public parameters PP, master private key MK, proxy update key UK and proxy grant key GK.

- **DO.Enc** $(PP, M, \mathcal{T}) \rightarrow CT$: The encryption algorithm takes public parameters PP, a message M, and an access structure $\mathcal{T}$ over the universe of attributes as input. It generates a ciphertext CT.
- **AA.KenGen** $(MK, U_{id}, S_{U_{id}}) \rightarrow (SK_{U_{id}}, A_{U_{id}}, B_{U_{id}})$: The key generation algorithm takes master private key MK, a unique user identity U_{id} and the corresponding attribute set $S_{U_{id}}$ as input. It outputs U_{id}'s corresponding private key $SK_{U_{id}}$, user's search key $A_{U_{id}}$, and user's search key $B_{U_{id}}$ in CS.
- **U.Dec** $(CT, SK_{U_{id}}) \rightarrow M$: The decryption algorithm take a ciphertext CT, and a private key $SK_{U_{id}}$ as input. If the set of attributes $S_{U_{id}}$ related to $SK_{U_{id}}$ satisfies the access structure $\mathcal{T}$ related to CT, then it successfully decrypt and output the message M
- **AA.ReKenGen** (PP, MK, γ, ΔL_{γ}, η, ΔL_{η}, UK, GK)$\rightarrow$(PP', MK', UK', GK', $RK\gamma$): The re-encryption key generation algorithm tekes public parameters PP, a master key MK, a set of attributes γ, the attribute in γ which is to be revoked for some users, and the corresponding revoked user list ΔL_{γ}, a set of attribute η, the attribute in η is to be granted for some users, and the corresponding granted user list ΔL_{η}, proxy update key UK, and proxy grant key GK as input. It generates the updated public parameters PP', the redefined master key MK', the redefined proxy update key UK', the proxy grant key GK', and the re-encryption key RK_{γ}.
- **CS.ReEnc** (γ, C_{γ}, rk)$\rightarrow RK_{\gamma}$: The re-encryption algorithm takes a set of attribute γ which some users be revoked, the ciphertext component $C_{\gamma} = \{C_x\}_{att(x)\in\gamma} \subset CT$, the re-encryption key RK_{γ} as input. It outputs the re-encryption ciphertext component $C_{\gamma}{}' = \{C_x^*\}_{att(x)\in\gamma}$.
- **PS.ReKey** $(U_{id}, S_{U_{id}}, ver_{U_{id}}, D_{U_{id}}, UK, \gamma) \rightarrow (S_{U_{id}}{}', ver_{U_{id}}{}', D_{U_{id}}{}')$: The key regeneration algorithm takes a unique user identity U_{id} and the corresponding attribute set $S_{U_{id}}$, version number $ver_{U_{id}}$, the private key component $D_{U_{id}} = \{D_i\}_{i\in\gamma} \subset SK_{U_{id}}$, proxy update key UK, a set of attributes γ, the attribute in γ which is to be revoked by some users as input. It outputs the updated user attribute set $S_{U_{id}}{}'$, version number $ver_{U_{id}}{}'$ and private key component $D_{U_{id}}{}' = \{D_i'\}_{i\in\gamma} \subset SK_{U_{id}}$.
- **PS.GrantAtt** $(U_{id}, PP, GK, \eta) \rightarrow (\eta_{U_{id}}, SK_{\eta_{U_{id}}})$: The attribute grant algorithm takes a unique user identity U_{id}, the public parameters PP, the proxy grant key GK, a set of attribute η, the attribute in η which is to be granted by some users as input. It outputs a set of attribute $\eta_{U_{id}}$ which is to be granted to the user U_{id} and the corresponding private key component $SK_{\eta_{U_{id}}} = \{D_i, D_i^*\}_{i\in\eta_{U_{id}}}$.
- **O.PreIndex** $(W)\rightarrow(E)$: The pre index generation algorithm for the data owner takes keyword set $W = \{w_1, \cdots, w_m\}$ as input. It outputs data owner's pre keywords index set $E = (E_1, \cdots, E_m)$.
- **CS.Index** $(E, B_{U_{id}}) \rightarrow (V)$: The index generation algorithm for the CS takes data owner's pre keywords index set $E = (E_1, \cdots, E_m)$ and data owner's search key in CS $B_{U_{id}}$ as input. It outputs CS's index parameter set $V = (V_1, \cdots, V_m)$.
- **O.PostIndex** $(V, A_{U_{id}}) \rightarrow (I_W)$: The post index generation algorithm for DO takes CS's index parameter set $V = (V_1, \cdots, V_m)$, his own search key $A_{U_{id}}$ as input. It outputs DO's post keywords index set $I_W = (I_{w_1}, \cdots, I_{w_m})$.
- **U. PreTrap** $(w, A_{U_{id}}) \rightarrow (T_w)$: The pre trapdoor generation algorithm for user takes a keyword w and his own private search key $A_{U_{id}}$ as input. It outputs user's pre trapdoor T_w.

- **CS. PostTrap** $(T_w, B_{U_{id}}) \rightarrow (k')$: The post trapdoor generation algorithm for CS takes user's pre trapdoor T_w and user's search key $B_{U_{id}}$ in CS as input. It outputs CS's post trapdoor k'.
- **CS. Test**$(I_W, k') \rightarrow \{1, 0\}$: The test algorithm for the CS takes post keywords index set I_W and post trapdoor k' as input. If the match is successful, the output is 1. Otherwise, the output is 0.

C. Security Definitions

Similar to most previous works, the CS is supposed to be "curious-but-honest" [13].

We consider the security model as two games between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Game 1 (IND-sCP-CPA security model). The adversary $\mathcal{A}$ is assumed to be an outsider attracter including the receiver.

Int. $\mathcal{A}$ declares an access structure $\mathcal{T}^*$.

Setup. $\mathcal{C}$ takes a security parameter λ and runs the **Setup** algorithm. It gives the public parameter PP to $\mathcal{A}$ and keeps the master key MK to itself.

Phase 1. $\mathcal{A}$ adaptively issues polynomial queries as follows.

- *private key query*. $\mathcal{A}$ submits an attribute set S, where S does not satisfy the access structure $\mathcal{T}^*$, to $\mathcal{C}$. The challenger returns the corresponding private key SK to $\mathcal{A}$.
- *update private key query*. $\mathcal{A}$ is allowed to issue queries for update private key SK for the attribute in γ which is to be revoked for some users. The challenger gives the updated private key SK'.

Challenge. $\mathcal{A}$ submits two equal length message M_0 and M_1. The challenger picks a random bit $b \in \{0, 1\}$ and encrypts M_b under $\mathcal{T}^*$. The challenger gives ciphertext CT^* to $\mathcal{A}$.

Phase 2. Repeat Phase 1 adaptively.

Guess. $\mathcal{A}$ outputs a guess b' of b and wins the game if $b' = b$.

The advantage of the $\mathcal{A}$ in this game is defined as $\left|\Pr[b' = b] - \frac{1}{2}\right|$.

Definition 4. The proposed scheme is IND-sCP-CPA secure if there is no polynomial time $\mathcal{A}$ who can win the above game with non-negligible advantage.

Game 2 (IND -CKA security model). The adversary $\mathcal{A}$ is assumed to be CS.

Setup. Repeat game 1's setup adaptively.

Phase 1. $\mathcal{A}$ adaptively issues polynomial following queries.

- H_1-*Query*. $\mathcal{A}$ can query the random oracle H_1.
- H_2-*Query*. $\mathcal{A}$ can query the random oracle H_2.
- *Trapdoor Queries*. $\mathcal{A}$ can ask any keyword's trapdoor.

Challenge. $\mathcal{A}$ submits two keywords w_0 and w_1 where the keywords w_0 and w_1's trapdoor have not been asked by $\mathcal{A}$. The challenger picks a random bit $b \in \{0, 1\}$ and creates w_b's trapdoor k to $\mathcal{A}$.

Phase 2. Repeated phase 1 adaptively.

Guess. $\mathcal{A}$ submits a guess b' of b. If $b' = b$, $\mathcal{A}$ wins the game and break our scheme.

Definition 5. In the random oracle model, the proposed scheme is IND-CKA secure if all polynomial time adversaries have at most a negligible advantage in the above game.

IV. Our Proposed Scheme

A. Detail Construction of Algorithms

AA defines the universe of attributes as $U = \{1, 2, \cdots, n\}$, the unique user identity $U_{id} \in \{0, 1\}^*$ and three hash functions:

- $H(\cdot)$: Maps an attribute to a random element of $\mathbb{G}_1$.
- $H_1(\cdot)$: Maps a strings in $\{0, 1\}^*$ to a random element of $\mathbb{G}_1$.
- $H_2(\cdot)$: Maps a random element of $\mathbb{G}_2$ to a random strings of $\{0, 1\}^l$.

AA.Setup $(\lambda, U) \rightarrow (PP, MK, UK, GK)$: The setup algorithm takes security parameter λ and attribute universe description $U = \{1, 2, \cdots, n\}$ as input. It first chooses two multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order $p(p > 2^{\lambda})$, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Then, $\forall i \in U$, it random chooses $Att_i \in Z_p^*$ and computes a public parameter component $T_i = g^{Att_i} \in \mathbb{G}_1$. And it randomly chooses $x \in Z_P$ as the search master key and three random numbers $\alpha, \beta, x \in Z_p$, let

$$PP = (\mathbb{G}_1, g, g^{\beta}, e(g,g)^{\alpha}, T_1, \cdots, T_n)$$

$$MK = (x, \beta, g^{\alpha}, Att_1, \cdots, Att_n)$$

In addition, defines the system version number $ver \in N$. The initial version number is set to $ver = 0$. Set a proxy re-encrypt key set as $rk = \{rk_i\}_{i \in U}$, and $rk_i = \{rk_{i,0}, \cdots, rk_{i,ver}\}$ is set of the proxy re-encrypt key under different version for attribute i. The initial value is set to $rk_{i,0} = 1$.

Let $L = \{L_i\}_{i \in U}$ represent the revoked user list set, where revocation list L_i represents the users list whose attribute i needs to be revoked, and $L^* = \{L_i^*\}_{i \in U}$ represent the granted user list set, where grant list L_i^* represents the users list set to whom the attribute i needs to be granted. The revocation list L_i may be empty, which means there is no user needs to be revoked for attribute i. So is grant list L_i^*.

Finally, define a set R which is used to reserve the private key component $(U_{id}, r_{U_{id}})$ later. The initial value is empty, $R = \phi$. For each $Att_i \in Z_p^*$, calculate $\frac{1}{Att_i} (\mathrm{mod}\, p)$ and output

The proxy update key $UK = (ver, rk = \{rk_i\}_{i \in U}, L = \{L_i\}_{i \in U})$;

The proxy grant key $GK = \left(R, \frac{1}{Att_1}, \cdots, \frac{1}{Att_n}, L^* = \{L_i^*\}_{i \in U}\right)$.

Do.Enc $(PP, M, \mathcal{T}) \rightarrow CT$: Similar to the encryption method in [18]. It inputs the public parameter PP, a message M and an access structure $\mathcal{T}$. The algorithm chooses a (k_x-1) -degree polynomial $q_x(\cdot)$ for each node x in the tree $\mathcal{T}$ in a top-down manner. The selected polynomial $q_x(\cdot)$ must satisfy the restriction that $q_x(0) = s$ if x is the root node in $\mathcal{T}$, otherwise $q_x(0) = q_{parent(x)}(indes(x))$, where s is randomly chosen from Z_p^*. It is worth noting that for a leaf node, because it does not have a child, so it selects constant polynomial $q_x(\cdot) = q_x(0) = q_{parent(x)}(indes(x))$. Let Ψ be the set of leaf nodes in $\mathcal{T}$, the ciphertext CT is computed as:

$$CT = (\mathcal{T}, \tilde{C} = M \cdot e(g,g)^{\alpha s}, C = g^{\beta s}, \forall x \in \Psi : C_x = g^{q_x(0) T_{att(x)}}, C_x^* = H(att(x))^{q_x(0)})$$

Here, the function $att(x)$ returns the attribute associated with the leaf node x and $att(x) \in U$. Note that, the hash function $H(\cdot)$ Maps an attribute to a random element of $\mathbb{G}_1$, so $H(att(x)) \in \mathbb{G}_1$.

AA.KenGen $(MK, U_{id}, S_{U_{id}}) \rightarrow (SK_{U_{id}}, A_{U_{id}}, B_{U_{id}})$: The key generation algorithm takes master private key MK, a unique user identity U_{id} and the corresponding attribute set $S_{U_{id}}$ as input. It firstly defines a user version number as the current system version number $ver_{U_{id}} = ver$.

Then it chooses a random $r_{U_{id}} \in Z_P$, and then chooses a random $r_i \in Z_P$ for each attribute $i \in S_{U_{id}}$. It outputs the private key as:

$$SK_{U_{id}} = \left(ver_{U_{id}}, D = g^{\frac{\alpha + r_{U_{id}}}{\beta}}, \forall i \in S_{U_{id}} : D_i = g^{\frac{r_{U_{id}}}{Att_i}} \cdot H(i)^{\frac{r_i}{Att_i}}, D_i^* = g^{r_i} \right)$$

It randomly chooses $\mu \in Z_P$ and set $A_{U_{id}} = \mu$ as user's search key, and computes $B_{U_{id}} = g^{\frac{x}{A_{U_{id}}}} = g^{\frac{x}{\mu}}$ as user's search key in CS.

U.Dec $(CT, SK_{U_{id}}) \rightarrow M$: Similar to the decryption method in [18]. The decryption algorithm first defines a recursive algorithm $DecNode\ (CT, SK_{U_{id}}, x)$, where x represents a node in $\mathcal{T}$. Then it is followed in a down-top manner.

- For each leaf node x, with $i = att(x)$, if $i \in S_{U_{id}}$, it computes: $DecNode\ (CT, SK_{U_{id}}, x) = \frac{e(D_i, C_x)}{e(D_i^*, C_x^*)} = e(g,g)^{r_{U_{id}} q_x(0)}$. Otherwise, it returns $\perp$;
- For each interior node x, Lagrange interpolation is used on at least k_x such $e(g,g)^{r_{U_{id}} q_{z_j}(0)}$ from its children $\{z_j\}$ to calculate $e(g,g)^{r_{U_{id}} q_x(0)}$.

Finally, for the root node $R_{\mathcal{T}}$ in $\mathcal{T}$, let $A = e(g,g)^{r_{U_{id}} q_{R_{\mathcal{T}}}(0)} = e(g,g)^{r_{U_{id}} s}$. The decryption can be computed by:

$$Dec\ (CT, SK_{U_{id}}) = \tilde{C} / (e(C, D)/A) = M$$

AA.ReKenGen $(PP, MK, \gamma, \Delta L_\gamma, \eta, \Delta L_\eta, UK, GK) \rightarrow (PP', MK', UK', GK', RK_\gamma)$: The re-encryption key generation algorithm takes public parameter PP, a master key MK, a set of attributes γ (the attribute in γ which is to be revoked for some users) and the corresponding revoked user list ΔL_γ, a set of attribute η (the attribute in η is to be granted for some users) and the corresponding granted user list ΔL_η, the proxy update key UK, and the proxy grant key GK as input.

If $\gamma \neq \emptyset$, for each attribute $i \in \gamma$, it chooses random $Att_i' \in Z_p^*$ as the new attribute key. Then performs the following action:

- Master key update. Replaces the Att_i in MK with Att_i', the rest of the parameters keeps unchanged;
- Proxy update key upodate. Replaces the ver in UK as $ver' = ver + 1$, calculates $rk_{i,ver'} = \frac{Att_i'}{Att_i}(p)$ and adds to the set $rk_i = \{rk_{i,0}, \cdots, rk_{i,ver}\}$. For other attribute $i \in U \setminus \gamma$, adds $rk_{i,ver'} = 1$ to the set $rk_i = \{rk_{i,0}, \cdots, rk_{i,ver}\}$ to get the updated $rk_i = \{rk_{i,0}, \cdots, rk_{i,ver}, rk_{i,ver'}\}$. Then adds the identity of users whose attribute need to be revoked in ΔL_γ to the corresponding revocation user list $L = \{L_i\}_{i \in U}$;
- Set re-encryption key $RK_\gamma = \{(i, rk_{i,ver'})\}_{i \in \gamma}$;
- Proxy grant key update. Replace the $\frac{1}{Att_i}$ in GK with $\frac{1}{Att_i'} (\mathrm{mod}\, p)$, the rest of the parameters keeps unchanged;
- Public parameter update. Calculate $T_i' = T_i^{rk_{i,ver'}} = g^{Att_i'}$ and replace the T_i in PP with T_i', the rest of the parameters keeps unchanged.

If $\eta \neq \emptyset$, add the identity of users in ΔL_η who need to be granted some attributes to the corresponding grant user list $L^* = \{L_i^*\}_{i \in U}$ in proxy grant key.

CS.ReEnc $(\gamma, C_\gamma, RK_\gamma) \rightarrow C_\gamma'$: The re-encryption algorithm takes a set of attribute γ, the attribute in γ which is to be revoked for some users, the ciphertext component $C_\gamma = \{C_x\}_{att(x) \in \gamma} \subset CT$, the re-encryption key RK_γ as input.

For each attribute $i \in \gamma$, find the corresponding leaf node x, with $i = att(x)$. Denote the universe of corresponding ciphertext component C_x set as $C_\gamma = \{C_x\}_{att(x) \in \gamma} \subset CT$. For each attribute $C_x \in C_\gamma$, calculate $C_x{}' = C_x{}^{rk_{i,ver'}}$ and output $C_\gamma' = \{C_x'\}_{att(x) \in \gamma}$.

PS.ReKey $(U_{id}, S_{U_{id}}, ver_{U_{id}}, D_{U_{id}}, UK, \gamma) \rightarrow (S_{U_{id}}{}', ver_{U_{id}}{}', D_{U_{id}}{}')$: The key regeneration algorithm takes a unique user identity U_{id} and the corresponding attribute set $S_{U_{id}}$, version number $ver_{U_{id}}$, the private key component $D_{U_{id}} = \{D_i\}_{i \in \gamma} \subset SK_{U_{id}}$, proxy update key UK, a set of attributes γ, the attribute in γ which is to be revoked by some users as input. Then perform the following actions:

- If the user has the latest version $ver_{U_{id}} = ver$, it outputs $\perp$ and exit;
- If it satisfies the condition that $\forall i \in \gamma$ and $U_{id} \notin L_i$, denotes the attribute set $S'_{U_{id}} = S_{U_{id}}$. For each $i \in S_{U_{id}} \cap \gamma$ and $U_{id} \in L_i$, denotes the attribute set $S'_{U_{id}} = S_{U_{id}} \setminus \{i\}$ and deletes the U_{id} from L_i;
- For each $i \in S_{U_{id}} \cap \gamma$, compute $D_i^{*\prime} = D_i^{*\left(rk_{i,(ver_{U_{id}}+1)} \cdots rk_{i,ver}\right)^{-1}}$ and replace the D_i^* with $D_i^{*\prime}$. Then update the user version number as $ver_{U_{id}}{}' = ver$.

PS.GrantAtt $(U_{id}, PP, GK, \eta) \rightarrow (\eta_{U_{id}}, SK_{\eta_{U_{id}}})$: The attribute grant algorithm takes a unique user identity U_{id}, public parameters PP, proxy update key UK, proxy grant key GK, a set of attribute η, the attribute in η is to be granted for some users as input. Then perform the following actions:

- If it satisfies the condition that $\forall i \in \eta$ and $U_{id} \notin L_i^*$, it outputs $\perp$ and exit;
- Then build an attribute set $\eta_{U_{id}}$ as the grant set for user U_{id}. The initial value $\eta_{U_{id}} = \emptyset$. For each $i \in \eta$, if $U_{id} \in L_i^*$, add attribute i to the attribute set $\eta_{U_{id}}$ and delete the U_{id} from L_i^*;
- For each $i \in \eta_{U_{id}}$, find the parameter $(U_{id}, r_{U_{id}})$ from the list R and randomly choose $r_i \in Z_P$. Then compute $D_i = g^{\frac{r_{U_{id}}}{Att_i}} \cdot H(i)^{\frac{r_i}{Att_i}}$, $D_i^* = g^{r_i}$, and define the grant private key component as $SK_{\eta_{U_{id}}} = \{D_i, D_i^*\}_{i \in \eta_{U_{id}}}$.

DO.PreIndex $(W) \rightarrow (E)$: The pre index generation algorithm for data owner takes the keyword set $W = \{w_1, \cdots, w_m\}$ as input.

For each keyword $w_i \in W$, it calculates $E_i = H_1(w_i)^{l_i} \in \mathbb{G}_1$, where $l_i \in Z_p^*$ is a random number.

Then, it outputs the data owner's pre keywords index set $E = (E_1, \cdots, E_m)$.

CS.Index $(E, B_{U_{id}}) \rightarrow (V)$: The index generation algorithm by CS takes data owner's pre keywords index set $E = (E_1, \cdots, E_m)$ and data owner's search key in CS $B_{U_{id}}$ as input.

For each $E_i \in E$, it computes $V_i = e(E_i, B_{U_{id}}) = e\left(H_1(w_i)^{l_i}, g^{\frac{x}{A_{U_{id}}}}\right)$.

Then, it outputs CS's index parameter set $V = (V_1, \cdots, V_m)$.

DO.PostIndex $(V, A_{U_{id}}) \rightarrow (I_W)$: The post index generation algorithm for data owner takes the CS's index parameter set $V = (V_1, \cdots, V_m)$, his own search key $A_{U_{id}}$ and the random parameter l_i which he choices before as input.

- For each $V_i \in V$, it computes

$$k_i = H_2\left(V_i^{\frac{A_{U_{id}}}{l_i}}\right) = H_2\left(e\left(H_1(w_i)^{l_i}, g^{\frac{x}{A_{U_{id}}}}\right)^{\frac{A_{U_{id}}}{l_i}}\right) = H_2(\, e(H_1(w_i), g)^x) \in \{0,1\}^l;$$

- set $I_{w_i} = (Q_i, [Q_i]_{k_i})$, where $[Q_i]_{k_i}$ denotes an encryption of a random number Q_i with the secret key k_i using a secure symmetric encryption algorithm, such as AES.

It builds the data owner's post keywords index set $I_W = \{I_{w_1}, \cdots, I_{w_m}\}$ and outputs.

U. PreTrap $(w, A_{U_{id}}) \rightarrow (T_w)$: The pre trapdoor generation algorithm for user takes as input a keyword w and his own search key $A_{U_{id}}$.

It calculates the user's pre trapdoor $T_w = H_1(w)^{A_{U_{id}}}$ and outputs.

CS. PostTrap $(T_w, B_{U_{id}}) \rightarrow (k')$: The post trapdoor generation algorithm for CS input the pre trapdoor T_w and the user's search key $B_{U_{id}}$ in CS.

It calculates the CS's post trapdoor $k' = H_2(e(T_w, B_{U_{id}})) = H_2\left(e\left(H_1(w)^{A_{U_{id}}}, g^{\frac{x}{A_{U_{id}}}}\right)\right) = H_2(\, e(H_1(w_i), g)^x)$ and outputs.

CS. Test(I_W, k')→{1, 0}: The test algorithm by CS takes post keywords index set $I_W = \{I_{w_1}, \cdots, I_{w_m}\}$ and post trapdoor $k' = H_2(e(H_1(w_i),g)^x)$ as input. It checks the following equation holds

$$\exists I_{w_i} = (Q_i, [Q_i]_{k_i}) \in I_W, \text{ such that } [Q_i]_{k'} = [Q_i]_{k_i}$$

If the equation holds, it outputs 1. Otherwise, it outputs 0.

B. Main Construction

System Setup. AA first asked about **AA.Setup** (λ, U)→(PP, MK, UK, GK) algorithm to get public parameter PP, master key MK, proxy update key UK, and proxy grant key GK. Then AA sends PP to CS and keeps UK, GK, MK secret.

Registration. AA to register every legal user in the system.

- Select a unique identity U_{id} and an attribute set $S_{U_{id}}$ to user;
- Call algorithm **AA.KenGen** $(MK, U_{id}, S_{U_{id}}) \rightarrow (SK_{U_{id}}, A_{U_{id}}, B_{U_{id}})$ to compute a private key $SK_{U_{id}}$, user's search key $A_{U_{id}}$, and user's search key $B_{U_{id}}$ in CS.
- Update the set $R = R \cup \{(U_{id}, r_{U_{id}})\}$ in $GK = \left(R, \frac{1}{Att_1}, \cdots, \frac{1}{Att_n}, L^* = \{L_i^*\}_{i \in U}\right)$;

Finally, AA transmits the tuple $(U_{id}, A_{U_{id}}, S_{U_{id}}, SK_{U_{id}})$ to new user, transmits GK to PS and transmits the tuple $(U_{id}, B_{U_{id}}, S_{U_{id}})$ to CS. CS adds the new user information tuple to the users information list.

Establishment of Index. DO first extracts a set of keywords $W = \{w_1, \cdots, w_m\}$ from the file to establish a keyword index.

- DO calls algorithm **DO.PreIndex**(W)→(E). It outputs DO's pre keywords index set $E = (E_1, \cdots, E_m)$. DO sends his identity U_{id} together with the pre keywords index set E to the CS.
- After receiving the request, CS first obtains the data owner's corresponding $B_{U_{id}}$ according to U_{id}. Then CS calls the algorithm **CS.Index** $(E, B_{U_{id}}) \rightarrow (V)$. It outputs CS's index parameter set $V = (V_1, \cdots, V_m)$. Then, CS transmits V to DO.

Table 2. File Storage Format in CS.

F_{id}	I_W	CT	$[DataFile]_k$

doi:10.1371/journal.pone.0167157.t002

- DO inputs his own private search key $A_{U_{id}}$ and the random parameter l_i which he choices before, and calls algorithm **O.PostIndex** $(V, A_{U_{id}}, l_i) \rightarrow (I_W)$. It outputs the DO's post keywords index set $I_W = (I_{w_1}, \cdots, I_{w_m})$.

File Upload. The file upload process is similar to the *D. Data Upload* process in literature [18]. The final document is stored in CS as Table 2.

Here, F_{id} represents the file number. $[DataFile]_k$ represents the encrypt file by a symmetric encryption key k. I_W represents the keywords index. CT represents the symmetric encryption key k's ciphertext which encrypted by our proposed algorithm **Do.Enc** $(PK, M = k, \mathcal{T}) \rightarrow CT$. Details of file upload process can be found in *D. Data Upload* in literature [18].

Attribute Alteration. If there is no need to change any user's attributes in the system, it outputs $\perp$ and exit.

If there have a set of attribute γ which some users be revoked, and a set of attribute η which some users be granted. We processes as follows.

- AA first calls the algorithm **AA.ReKenGen** (PP, MK, γ, ΔL_γ, η, ΔL_η, UK, GK)→(PP', MK', UK', GK', RK_γ) to obtain updated public parameter $PP = PP'$, master key MK, proxy update key $UK = MK'$, proxy grant key $GK = GK'$, and re-encryption key RK_γ. Then it sends PP, γ, RK_γ to CS, sends UK, GK to PS and keeps MK, UK, Gk secret.
- On receiving PP, CS publishes it.
- After receive the re-encryption key RK_γ, CS calls algorithm **CS.Enc** (γ, C_γ, RK_γ)→C_γ' and updates the corresponding ciphertext.

The following steps are performed when a user needs to search for a file.

Trapdoor Generation. First, the user set a search keyword w. Then he calls the algorithm **U. PreTrap** $(w, A_{U_{id}}) \rightarrow (T_w)$. It outputs the user's pre trapdoor T_w.

Updating Attribute and Private Key.

- User U_{id} sends his parameters $(U_{id}, S_{U_{id}}, ver_{U_{id}}, D_{U_{id}})$ to PS.
- PS first calls algorithm **PS.ReKey** $(U_{id}, S_{U_{id}}, ver_{U_{id}}, D_{U_{id}}, UK) \rightarrow (S_{U_{id}}', ver_{U_{id}}', D_{U_{id}}')$. It outputs updated user attribute set $S_{U_{id}}'$, version number $ver_{U_{id}}'$ and private key component $D_{U_{id}}' = \{D_i'\}_{i \in \gamma} \subset SK_{U_{id}}$.
- Then PS calls the algorithm **PS.GrantAtt** $(U_{id}, PP, GK) \rightarrow (\eta_{U_{id}}, SK_{\eta_{U_{id}}})$. It outputs a set of attribute $\eta_{U_{id}}$ which is to be granted to user U_{id} and the corresponding private key component $SK_{\eta_{U_{id}}} = \{D_i, D_i^*\}_{i \in \eta_{U_{id}}}$.
- It sets the parameters $S_{U_{id}} = S_{U_{id}}^{'} \cup \eta_{U_{id}}$.
- PS returns parameters $(U_{id}, S_{U_{id}}, ver_{U_{id}}', D_{U_{id}}, SK_{\eta_{U_{id}}})$ to user and send $(U_{id}, S_{U_{id}})$ to CS.
- User updates his own parameters $S_{U_{id}}$ and $SK_{U_{id}}$. CS updates tuple $(U_{id}, B_{U_{id}}, S_{U_{id}})$ for user U_{id}'s attribute in the users information list.

Search the File by CS. U_{id} sends his trapdoor $T_w = H_1(w)^{A_{U_{id}}}$ and his unique identity U_{id} to CS. CS performs the following action.

- according to user's identity U_{id} CS finds the corresponding user attribute set $S_{U_{id}}$ and user's search key $B_{U_{id}}$ in CS from user information list tuple $(U_{id}, B_{U_{id}}, S_{U_{id}})$.
- For the trapdoor $T_w = H_1(w)^{A_{U_{id}}}$ and user's search key $B_{U_{id}}$, CS calls algorithm **CS. Post-Trap** $(T_w, B_{U_{id}}) \rightarrow (k')$. It outputs CS's post trapdoor k'.
- According to attributes set $S_{U_{id}}$, CS has to search documents by performing the *Step3*: *search the data by the cloud server* process in literature [18].
- For all documents in the files collection that the user can decrypt, matches the keyword trapdoor with the keywords index, to find the user's interested files in the document.
- According to the search parameter k', it runs algorithm **CS. Test**$(I_W, k') \rightarrow \{1, 0\}$. If the output is 1, it returns the corresponding CT and $\{DataFile\}_k$ to the user.

File Decryption. User to decrypt ciphertext by calling decryption algorithm **U.Dec** $(CT, SK_{U_{id}}) \rightarrow M = k$. User to further decrypt the symmetric ciphertext $\{DataFile\}_k$ to get the document $\{DataFile\}$.

Similar to literature [18], we can also perform most of the calculation process by PS.

User Revocation. Our scheme by removing user's search key $B_{U_{id}}$ in CS to achieve user identity revocation. Because if CS to remove user's $B_{U_{id}}$, the user will not be able to successfully search files.

C. Flowchart of Our Proposed Scheme

We set a legitimate user U_{id} first as a data owner to upload their own data, and then as a user access to the content of the interest files. The flowchart of our Fig 2(a), 2(b), 2(c), 2(d) and 2(e) respectively gives the process of system setup, new user registration, file upload, system version upgrade and ciphertext update, file search by user of our scheme.

V. Security

A. Security Analysis

First of all, we analyze our scheme. There are six entities in the project: AA, PS, CS, DO, U.

AA is responsible for establishing the program, and we set it to be fully trusted.

Our PS is subordinate to authority. In order to reduce the computing load of AA and ensure the efficiency of program, we grant a lot of functions to PS. One of the most important functions is to grant user's attribute and the corresponding private key, which makes our PS must be trusted. If we think about the problem of malicious PS, we have to leave granted rights to AA. Only AA can execute the private key grant rights, which can improve the security of scheme to a certain extent, but it will reduce efficiency of scheme. It introduces a new model. We aim to study the integrity of our proposed scheme, which has not made a number of analysis to this new model.

DO has no difference from other users in set the private key in addition to having the data files to be uploaded. It is to say that keyword search process of DO is equivalent to a general user. Due to the users access permissions are different according to their own attribute set. Some users may want to access more data files beyond their access permissions. So one of the attack models we consider is derived from a malicious user. He may also be a legitimate user. We will show that our scheme is secure against this attack model.

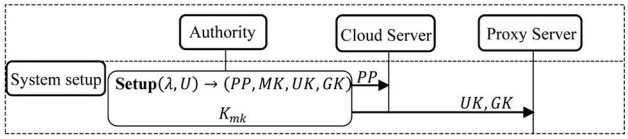

Fig 2. (a) System setup

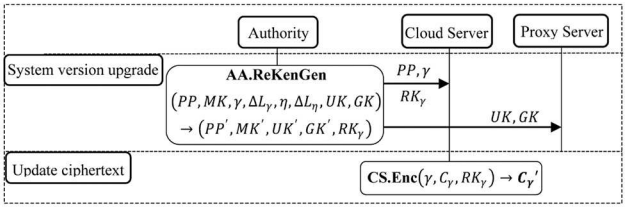

Fig 2. (d) System version upgrade and ciphertext update

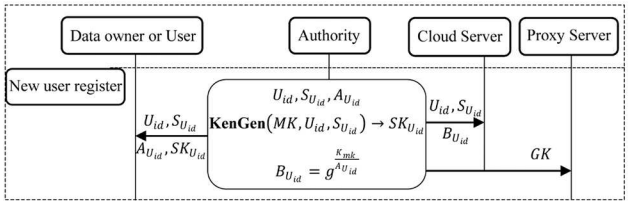

Fig 2. (b) New user register

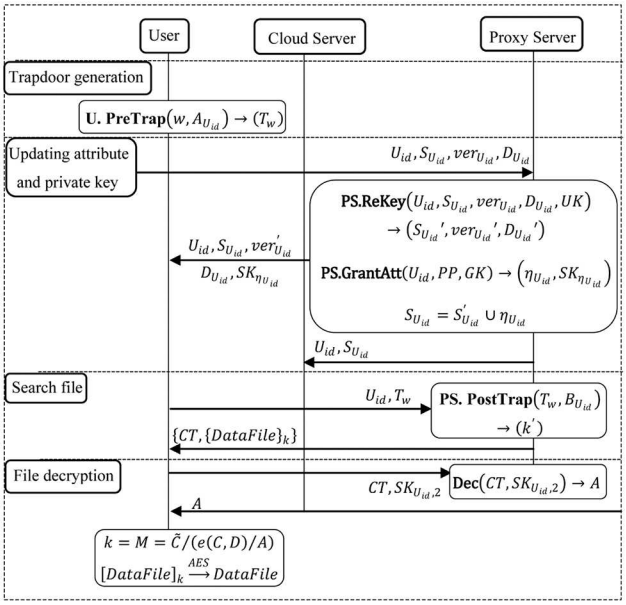

Fig 2. (e) File search by user

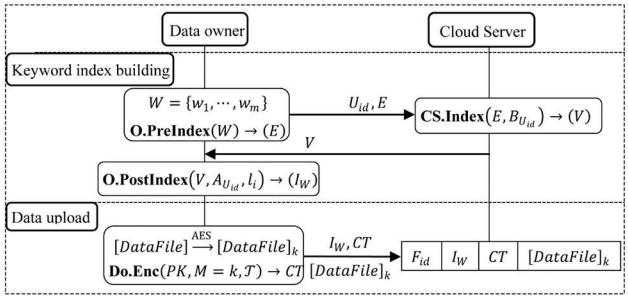

Fig 2. (c) Files upload

Fig 2. Flowchart of Our Proposed Scheme.

doi:10.1371/journal.pone.0167157.g002

CS is an outsourced server. As in most articles, we assume CS is "curious-but-honest" [13]. It is to say that CS is curious about the encrypted data contents or the received messages, but will execute correctly the proposed tasks. It might be interested in the content of user search, so another attack models we consider is derived from a malicious CS.

B. Attack Model 1 (IND-sCP-CPA Security Model)

The adversary $\mathcal{A}$ is assumed to be an outsider attacker including the users in the system.

Through the establishment of a security game model, we reduce the security of our scheme to Bethencourt's scheme [3]. According to the proof of reference [3] in its appendix A (Bethencourt's scheme is IND-sCP-CPA secure), our scheme is also IND-sCP-CPA secure in the attack model 1. The proof procedure is as follows.

Theorem 5 Suppose that the Bethencourt's scheme is IND-sCP-CPA secure, then our scheme is also IND-sCP-CPA secure in the attack model 1.

Proof. We consider a simulator $\mathcal{S}'$ of Bethencourt's scheme, a simulator $\mathcal{S}$ of our scheme and a polynomial-time adversary $\mathcal{A}$ of our scheme. It is noteworthy that the simulator $\mathcal{S}$ of our scheme has another identity who is also an adversary $\mathcal{A}'$ of Bethencourt's scheme. Suppose that $\mathcal{A}$ of our scheme is able to distinguish a valid ciphertext from a random element with advantage ε. We build a simulator $\mathcal{S}$ (namely $\mathcal{A}'$ of Bethencourt's scheme) that can attack Bethencourt's scheme with the same advantage. The simulation proceeds as follows.

Int. $\mathcal{A}$ declares an access structure $\mathcal{T}^*$, which he wishes to be challenged upon. The simulator $\mathcal{S}$ declares the same access structure.

Setup. The simulator $\mathcal{S}'$ takes a security parameter λ and runs the **Setup** algorithm of Bethencourt's scheme. It gives the public parameter $PP' = (\mathbb{G}_1, g, g^\beta, e(g,g)^\alpha)$ to the simulator $\mathcal{S}$. After receive the public parameter PP', the simulator $\mathcal{S}$ randomly chooses $Att_i \in Z_p^*$ for $\forall i \in U$ as the attribute parameter. Then for all $i \in U, j = 1, 2, \cdots, ver$, it randomly chooses $rk_{i,j} \in Z_p^*$ and the public parameter generated as follows.

$$PP = (\mathbb{G}_1, g, g^\beta, e(g,g)^\alpha, T_1 = g^{Att_1}, \cdots, T_n = g^{Att_n})$$

$$MK' = (ver, Att_1, \cdots, Att_n, rk = \{rk_i\}_{i \in U})$$

Then, it send the public parameter PP to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively issues polynomial following queries.

- ***private key query***. $\mathcal{A}$ submits a set of attributes S where S does not satisfy the access structure $\mathcal{T}^*$ to the simulator $\mathcal{S}$. The simulator $\mathcal{S}$ submits the same attributes to the simulator $\mathcal{S}'$. Then the simulator $\mathcal{S}$ can get the $SK' = (D = g^{\alpha+r/\beta}, \forall i \in S : D_i' = g^r H(i)^{r_i}, D_i^{*'} = g^{r_i})$ from the simulator $\mathcal{S}'$. The simulator $\mathcal{S}$ calculates as follows. For all $i \in S, D_i = (D_i')^{1/Att_i}$. The $\mathcal{A}$ is given private key $SK = (D = g^{\alpha+r/\beta}, \forall i \in S : D_i = (D_i')^{1/Att_i}, D_i^* = D_i^{*'})$.
- ***update private key query***. $\mathcal{A}$ is allowed to issue queries for update private key SK for the attribute in γ which is to be revoked for some users. $\mathcal{A}$ submits a part of the private key $\hat{SK} = (\forall i \in S : D_i, D_i^*)$ he asked before where $S \cap \gamma \neq \phi$.
  1. For all attribute $i \in \gamma$, the simulator $\mathcal{S}$ randomly chooses $rk_i \in Z_p^*$ and maintains the update key $\{rk_i\}_{i \in \gamma}$.
  2. For all attribute $i \in S \cap \gamma$ in $\hat{SK}$, the simulator $\mathcal{S}$ calculates $D_i' = D_i^{\frac{1}{rk_i}}$ and keeps other parameters unchanged. Then returns the update private key $\hat{SK}'$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two messages M_0 and M_1 on which he wishes to be challenged upon. $\mathcal{S}$ outputs the same messages to $\mathcal{S}'$. Then $\mathcal{S}'$ flips a random coin $b \in \{0, 1\}$, and encrypts M_b with access structure $\mathcal{T}^*$. $\mathcal{S}'$ sends the ciphertext $CT' = (\mathcal{T}^*, \tilde{C} = M_b \cdot e(g,g)^{\alpha s}, C = g^{\beta s}, \forall x \in \Psi : C_x' = g^{q_x(0)}, C_x^{*'} = H(att(x))^{q_x(0)})$ to $\mathcal{S}$. The simulator $\mathcal{S}$ calculates CT^* as follows. For all $x \in \Psi, C_i = (C_x')^{T_{att(x)}}$. $\mathcal{A}$ is given attribute key
$CT^* = (\mathcal{T}^*, \tilde{C} = M \cdot e(g,g)^{\alpha s}, C = g^{\beta s}, \forall x \in \Psi : C_i = (C_x')^{T_{att(x)}}, C_x^* = C_x^{*'})$.

Phase 2. Repeated phase 1 adaptively.

Guess. $\mathcal{A}$ submits a guess b' of b. $\mathcal{S}$ outputs the guess b' to indicate that it was given the CT'. If $\mathcal{A}$ is able to distinguish the valid ciphertext with advantage $|\Pr[b' = b] - \frac{1}{2}| = \varepsilon$. We build the simulator $\mathcal{S}$ that can distinguish the valid ciphertext in Bethencourt's scheme with the same advantage.

C. Attack Model 2 (IND-CKA Security Model)

The adversary $\mathcal{A}$ is assumed to be CS.

We will prove that our scheme of semantic security for keywords trapdoor. Notice that in the search process of our scheme the public parameter is g, the private key of the user is $A_{U_{id}} = \mu$, the private key of CS is $B_{U_{id}} = g^{x/\mu}$, the master private key of the attribute authority is

$K_{mk} = x$. We assume that $\mathcal{A}$ is a malicious CS, then the public parameters related to the search process that it can get are $(g, B_{U_{id}} = g^{x/\mu})$.

Theorem 6. Assuming the BDH (Bilinear Diffie-Hellman) assumption was founded. Then our scheme has the IND-CKA security in the random oracle model.

Proof. We consider a chosen-keywords-attack polynomial-time adversary $\mathcal{A}$ and a simulator $\mathcal{S}$.

Suppose that $\mathcal{A}$ is able to correctly distinguish keywords with advantage ε. We build a simulator $\mathcal{S}$ that can solve the BDH problem with at least $\varepsilon' = 2\varepsilon/\hat{e}q_T q_{H_2}$, Where $\hat{e}$ is the base of the natural logarithm, $q_T > 0$ is the number of pre trapdoor queries, $q_{H_2} > 0$ is the number of hash queries.

Int. The simulator $\mathcal{S}$ runs $\mathcal{A}$ and receives a BDH challenge. It first chooses two multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. $\mathcal{S}$ is given $g \in \mathbb{G}_1$, as well as $u_1 = g^{\alpha}$, $u_2 = g^{\beta}$, $u_3 = g^{\gamma}$ for some random $\alpha,\ \beta,\ \gamma \in \mathbb{Z}_p$. $\mathcal{S}$'s goal is to get $e(g,g)^{\alpha\beta\gamma} \in \mathbb{G}_2$.

Setup. The simulator $\mathcal{S}$ announces the public parameter $(g, B_{U_{id}} = u_3)$ with the implicit assumption that $K_{mk} = x = \beta \cdot \gamma,\ A_{U_{id}} = \mu = \beta$. According to the above settings, we can calculate that $B_{U_{id}} = g^{x/\mu} = g^{\gamma} = u_3$.

Phase 1. $\mathcal{A}$ adaptively issues polynomial following queries.

- ***H_1*-Query:** $\mathcal{A}$ can always ask the random oracle H_1 of any keyword $w_i \in \{0, 1\}^*$. $\mathcal{S}$ answers the questions of $\mathcal{A}$ and records the results of each answer.
  If $\mathcal{A}$ submits a keyword $w_i \in \{0, 1\}^*$ that has not been asked, $\mathcal{S}$ does the following.
  1. $\mathcal{S}$ generates a random coin $c_i \in \{0, 1\}$, so that $\Pr[c_i = 0] = 1/(q_T + 1)$.
  2. $\mathcal{S}$ picks a random element $a_i \in Z_p^*$. If the coin $c_i = 0$, $\mathcal{S}$ computes $h_i = u_1 \cdot g^{a_i} \in \mathbb{G}_1$. If $c_i = 1$, $\mathcal{S}$ computes $h_i = g^{a_i} \in \mathbb{G}_1$.
  3. $\mathcal{S}$ adds the tuple (w_i, h_i, a_i, c_i) to the list H_1-list, and returns $H_1(w_i) = h_i$ to $\mathcal{A}$.
     If $\mathcal{A}$ submits a query w_i that has been asked, then $\mathcal{S}$ finds the tuple (w_i, h_i, a_i, c_i) in the H_1-list and responds $H_1(w_i) = h_i \in \mathbb{G}_1$ to $\mathcal{A}$.
- ***H_2*-Query:** $\mathcal{A}$ can always ask the random oracle H_2 of any $t_i \in \mathbb{G}_2$. $\mathcal{S}$ answers the questions of $\mathcal{A}$ and records the results of each answer.
  If $\mathcal{A}$ submits a $t_i \in \mathbb{G}_2$ that has not been asked. $\mathcal{S}$ chooses a random number $H_2(t_i) = V_i \in \{0, 1\}^{\log p}$ and adds the tuple (t_i, V_i) to the list H_2-list. Then it returns $H_2(t_i) = V_i$ to $\mathcal{A}$.
  If $\mathcal{A}$ submits a query t_i that has been asked, then $\mathcal{S}$ finds the tuple (t_i, V_i) in the H_2-list and responds $H_2(t_i) = V_i$ to $\mathcal{A}$.
- **Pre-trapdoor queries:** $\mathcal{A}$ can also ask the pre-trapdoor of any keyword $w_i \in \{0, 1\}^*$. $\mathcal{S}$ answers the questions of $\mathcal{A}$ as folloes.
  1. For a keyword $w_i \in \{0, 1\}^*$, $\mathcal{S}$ executes H_1-Query to get a tuple (w_i, h_i, a_i, c_i).
  2. If $c_i = 0$, $\mathcal{S}$ declares a failure and ends the game.
  3. If $c_i = 1,\ H_1(w_i) = h_i = g^{a_i} \in \mathbb{G}_1$. $\mathcal{S}$ generates $T_{w_i} = u_2{}^{a_i} = (g^{a_i})^{\beta}$ and returns T_{w_i} to $\mathcal{A}$ as response for the query.

Challenge. $\mathcal{A}$ submits two keywords w_0 and w_1 where the keywords w_0 and w_1's trapdoor had not asked by $\mathcal{A}$.

- $\mathcal{S}$ initiates H_1-Query twice to obtain $h_0, h_1 \in \mathbb{G}_1$, where $H_1(w_0) = h_0$, $H_1(w_1) = h_1$. If $c_0 = 1$ and $c_1 = 1$, then $\mathcal{S}$ reports a failure and terminates.
- Otherwise, we know at least one of c_0 and c_1 is equal to 0. If $c_0 = 0$ and $c_1 = 0$, $\mathcal{S}$ picks randomly a bit $b \in \{0, 1\}$.
- $\mathcal{S}$ picks a random element $k \in \{0, 1\}^{log\ p}$, and return $\{u_3, k\}$ to $\mathcal{A}$ as a response, where k imitates the post trapdoor in our proposed scheme. Note that, if $\mathcal{A}$ has an advantage in answer the above question. We have the implied settings:

$$\begin{aligned} k &= H_2(e(T_w,\ B_{U_{id}})) \\ &= H_2(e(\ H_1(w_i)^{\beta},\ u_3)) \\ &= H_2(e(\ H_1(w_i)^{\beta},\ u_3)) \\ &= H_2(e(g^{(\alpha+a_b)\ \beta},\ g^{\gamma})) \\ &= H_2(e(g,\ g)^{\ \beta\gamma(\alpha+a_b)}) \end{aligned}$$

Phase 2. Repeated phase 1 adaptively.

Guess. $\mathcal{A}$ submits a guess b' of b. If $b' = b$, $\mathcal{A}$ wins the game and break our scheme.

Correctness Analyses. In the above simulation scheme, if the adversary can break the game and distinguish the keyword with a non negligible probability that means that the random element k it chooses is $H_2(e(g,\ g)^{\ \beta\gamma(\alpha+a_b)})$. Then $\mathcal{S}$ can compute that $\frac{k}{u_2 u_3 g^{a_b}} = \mathrm{e}(g,g)^{\alpha\beta\gamma}$ which means it solves the DDH problem.

Probability Analyses. We can prove that if $\mathcal{A}$ can win the game with a non negligible probability ε, then $\mathcal{S}$ can solve the BDH problem with the probability at least $2\varepsilon/eq_T q_{H_2}$. That process in detail in [36].

Because of the BDH assumption that the BDH problem is difficult, so the probability $2\varepsilon/eq_T q_{H_2}$ is negligible. That is, our scheme is safe.

Taking attack model 2 (a selected keyword attack model from the cloud server) as an example, we give the specific flow chart of the game process in Fig 3.

VI. Performance Analysis and Comparison

A. Performance Analysis

The time complexity of our scheme. In the **Setup** phase, a public parameter and master key are generated. At this stage, the total number of attributes is defined as n. An exponentiation operation in $\mathbb{G}_1$ or $\mathbb{G}_2$ is defined as e. A pairing operation is defined as p. The time complexity of generating *PP*, *MK*, *UK*, *GK* is $(2 + 2n)\mathrm{e} + p$, 0, 0, *ne* respectively. We calculate the total time complexity of **Setup** is $(2 + 3n)e + p$.

In algorithm **Encrypt** for d_2 number of attributes that associated with access structure. In order to compute *CT*, the user needs to run $(2+2d_2)e + p$ operations. So the time complexity of Encrypt is $(2 + 2d_2)e + p$.

When generating the private key for a user with number of attributes d_1, AA needs to run $(2 + 3d_1)\ e$ addition operations in order to compute *SK*. So the time complexity of **KeyGeneration** algorithm is $(2 + 3d_1)\ e$.

In algorithm **re encryption** for d_3 number of attributes that ciphertext needs to update, CS needs to run d_3e addition operations in order to update ciphertext component. So the time complexity of **re encryption** algorithm is d_3e.

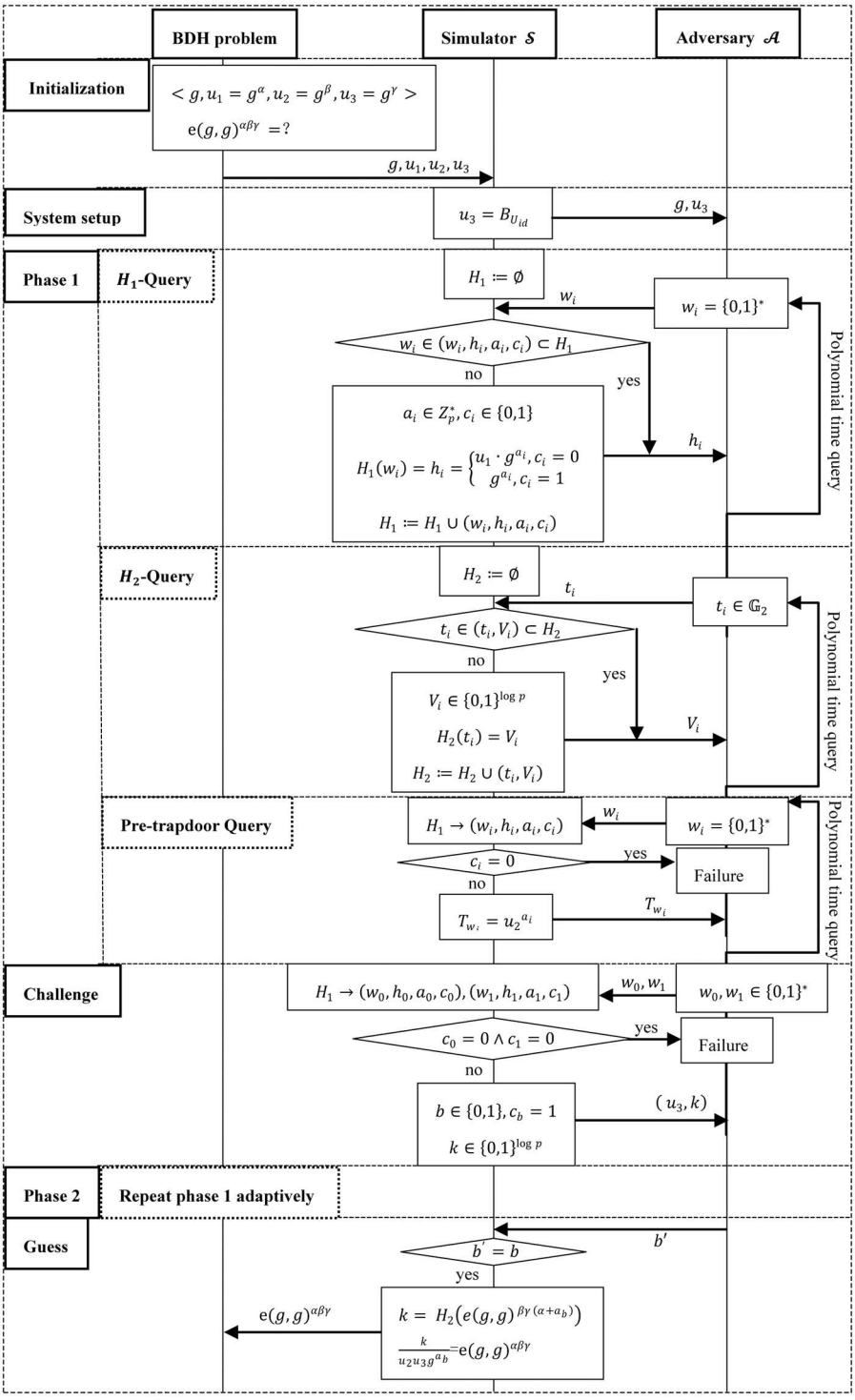

Fig 3. Flowchart of attack model 2.

doi:10.1371/journal.pone.0167157.g003

In algorithm **private key re generation** for d_4 number of attributes that a user needs to update, the PS needs to run d_4e addition operations in order to update ciphertext component. So the time complexity of **private key re generation** algorithm is d_4e.

In algorithm **attribute grant** for d_3 number of attributes that a user needs to granted, PS needs to run d_3e addition operations in order to compute the corresponding *SK*. So the time complexity of **attribute grant** algorithm is d_3e.

In algorithm **pre trapdoor** generation for a keyword *w*, the user needs to run an exponentiation operation in order to hide the keyword. So the time complexity of **pre trapdoor** algorithm is *e*.

In algorithm **post trapdoor** generation for CS, the user needs to run a pairing operation. So the time complexity of **post trapdoor** algorithm is *p*.

In algorithm **decryption** for d_6 number of user's attributes satisfying an access structure, the data owner needs to run $2e + (1 + d_6)p$ addition operations in order to compute the message *M*. So the time complexity of **decryption** algorithm is $2e + (1 + d_6)p$.

B. Comparison

We compare the computational complexity of our scheme with the existing schemes presented in [3, 13, 18] for the specific process in Table 3.

C. Simulation and Evaluation

In order to evaluate the performance of our CP-ABE construction, we test the runtime of the core algorithms Key Generation, Encryption and Decryption by user with different number of attributes. Fig 4 shows the test result. The implementation uses the Pairing Based Cryptography (PBC) library [37]. We can clearly see from Fig 4 that the key generation time and the encryption time increase with the number of attributes linearly, and the decryption time keeps

Table 3. Comparisons of Computational Complexity.

Schemes	[3]	[18]	[13]	Ours
PP	2e+p	2e+p	$(2 + 2n)$e+p	$(2 + 2n)$e+p
SK	$(2 + 3d_1)$e	$(2 + 3d_1)$e	$(2 + d_1)$e	$(2 + 3d_1)$e
CT	$(2 + 2d_2)$e+p	$(2 + 2d_2)$e+p	$(1 + 2d_2)$e+p	$(2 + 2d_2)$e+p
CT Update	✗	✗	d_3e	d_3e
SK Update	✗	✗	d_4e	d_4e
SK Grant	✗	✗	d_3e	d_3e
T_r by user	✗	e	✗	e
T_r by CS	✗	p	✗	p
DK	$2e+(1 + d_6)$p	$2e+(1 + d_6)$p	$(1 + 2d_6)$e+$(2 + 2d_6)$p	$2e+(1 + d_6)$p

e: an exponentiation operation in $\mathbb{G}_1$ or $\mathbb{G}_2$;
p: a pairing operation;
d_1: number of attributes that a user possess;
d_2: number of attributes that associated with access structure;
d_3: the number of attributes that ciphertext need to update;
d_4: number of attributes that a user needs to update;
d_5: number of attributes that a user needs to grant;
d_6: the number of user's attributes satisfying an access structure;
✗: there is no corresponding function or process in the literature.

doi:10.1371/journal.pone.0167157.t003

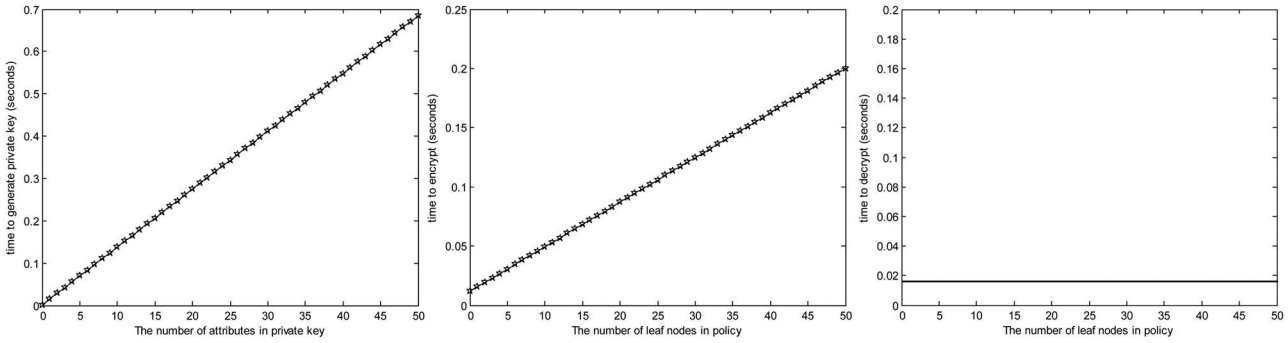

Fig 4. (a) Key generation time

Fig 4. (b) Encryption time

Fig 4. (c) Decryption time for user

Fig 4. The performance of CP-ABE.

doi:10.1371/journal.pone.0167157.g004

constant. This result is in agreement with our time complexity analysis in section Security and Performance analysis.

VI. Application

Our scheme is well suited for applications in cloud computing environments. Take search engine file management system for example. Firstly, users can become legitimate users by registered members. After the successful landing of legitimate users, users can not only search for documents of interest, but also upload local files to server.

On the one hand because of the excessive number of users and documents, the system through the outsourcing of data files to a CS.

On the other hand because the grade of membership system of the operating construction, making part of the document can only download by some VIP members. In order to facilitate the management of the system, the system can set up an interior PS to help manage user membership grade and duration.

Ordinary users can become VIP users by way of payment. The process of granting the VIP attribute does not require the system upgrade and the update of the ciphertext. AA only to issue a grand command to the PS. When a user access, PS verify that the user is required to grant the attribute according to the identity. To user who needs to be granted attributes, PS will grant the attribute and private key to the corresponding user in time.

Once the VIP attribute is invalid or expires, AA will update the system in a timely manner and send update command to PS.

VII. Conclusion

In this paper, we propose an efficiently multi-user searchable encryption with attribute revocation and grant function for cloud storage.

- In the first scenario, we propose a CP-ABE scheme with attribute revocation and grant. Our scheme can not only support a single user attribute revocation or grant, but also to some users to grant or revoke a set of attributes.
- In the second scenario, we propose a multi user search scheme based on a single keyword. As we focus on the user attribute update instead of the keyword search in this paper. Aiming at the problem of conjunctive keyword search is a direction that we continue to research.
- In addition, the lazy update of the user's attribute and the private key increases the efficiency of the scheme.

- Since PS in our scheme has the permissions granted attributes, in order to prevent a malicious PS to the user to grant a new attribute, we ask our PS must be honest and strict implementation of the tasks assigned by attribute authority. In other words, PS in strict accordance with the grant list to verify whether the user needs to grant attributes. Aiming at the problem of PS malicious attacks is another direction that we continue to research.

Supporting Information

S1 Appendix.
(DOCX)

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, Research Foundation of Education Department of Shaanxi Province of China under grants 2013JK1142. Thanks also go to the anonymous reviewers for their useful comments.

Author Contributions

Methodology: XZ SW.

Software: XZ YZ.

Validation: SW XZ.

Writing – original draft: SW XZ YZ.

Writing – review & editing: SW XZ YZ.

References

1. Sahai A, Waters B. Fuzzy Identity-Based Encryption: Springer Berlin Heidelberg; 2005. 457–73 p.
2. Goyal V, Pandey O, Sahai A, Waters B, editors. Attribute-based encryption for fine-grained access control of encrypted data. Proceedings of the 13th ACM conference on Computer and communications security; 2006.
3. Bethencourt J, Sahai A, Waters B, editors. Ciphertext-Policy Attribute-Based Encryption. IEEE Symposium; 2007: Security and privacy; 2008.
4. Emura K, Miyaji A, Nomura A, Omote K, Soshi M. A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length. International Journal of Applied Cryptography. 2009; 5451 (1):13–23.
5. Attrapadung N, Imai H, editors. Attribute-Based Encryption Supporting Direct/Indirect Revocation Modes. Ima International Conference on Cryptography and Coding; 2009 Dec 15–17; Cirencester, UK: Proceedings; 2009.
6. Martinez-Vara P, Barranco JS, IDLSG S, Munoz-Lopez J, Torres-Rodriguez MA, Xique RS, et al. Ciphertext-Policy Attribute-Based Threshold Decryption with Flexible Delegation and Revocation of User Attributes (extended version). Centre for Telematics & Information Technology University of Twente. 2009; 13(4):325–9.
7. Wu Q, Miao Z. Adaptively Secure Attribute-Based Encryption Supporting Attribute Revocation. Wireless Communication Over Zigbee for Automotive Inclination Measurement China Communications. 2012; 9 (9):22–40.
8. Zhang Y, Chen X, Li J, Li H, Li F, editors. FDR-ABE: Attribute-Based Encryption with Flexible and Direct Revocation. International Conference on Intelligent NETWORKING and Collaborative Systems; 2013: IEEE Computer Society; 2013.

9. Qiuxin. A Generic Construction of Ciphertext-Policy Attribute- Based Encryption Supporting Attribute Revocation. Wireless Communication Over Zigbee for Automotive Inclination Measurement China Communications. 2014; 11(A01):93–100.
10. Yu S, Wang C, Ren K, Lou W, editors. Attribute based data sharing with attribute revocation. ACM Symposium on Information; 2010 Apr; Beijing, China: Computer and Communications Security.
11. Chen JH, Wang YT, Chen KF. Attribute-Based Key-Insulated Encryption. Journal of Information Science & Engineering. 2011; 27(2):437–49.
12. Li Q, Feng D, Zhang L, editors. An attribute based encryption scheme with fine-grained attribute revocation. Global Communications Conference (GLOBECOM); 2012: IEEE.
13. Naruse T, Mohri M, Shiraishi Y. Provably secure attribute-based encryption with attribute revocation and grant function using proxy re-encryption and attribute key for updating. Human-centric Computing and Information Sciences. 2015; 5(1):1–13.
14. Bao F, Deng RH, Ding X, Yang Y. Private Query on Encrypted Data in Multi-user Settings. Lecture Notes in Computer Science. 2008; 4991:71–85.
15. Bringer J, Chabanne H, Kindarji B, editors. Error-tolerant searchable encryption. IEEE International Conference on Communications; 2009: IEEE.
16. Rhee HS, Park JH, Susilo W, Dong HL. Trapdoor security in a searchable public-key encryption scheme with a designated tester. Journal of Systems & Software. 2010; 83(5):763–71.
17. Hu C, Liu P. An Enhanced Searchable Public Key Encryption Scheme with a Designated Tester and Its Extensions. Journal of Computers. 2012; 7(3):716–23.
18. Lv Z, Zhang M, Feng D, editors. Multi-user Searchable Encryption with Efficient Access Control for Cloud Storage. IEEE International Conference on Cloud Computing Technology and Science; 2014: IEEE.
19. Yang Y, Lu H, Weng J, editors. Multi-User Private Keyword Search for Cloud Computing. IEEE International Conference on Cloud Computing Technology and Science; 2011 Nov 29-Dec; Athens, Greece: Cloudcom; 2011.
20. Liu Z, Wang Z, Cheng X, Jia C, Yuan K, editors. Multi-user Searchable Encryption with Coarser-Grained Access Control in Hybrid Cloud. International Conference on Emerging Intelligent Data and Web Technologies; 2013: IEEE Computer Society; 2013.
21. Jian Y, Yang D, editors. An agent-based searchable encryption scheme in mobile computing environment. International Conference on Computing, Communication and Networking Technologies; 2014: IEEE Computer Society.
22. Wang Q, Zhu Y, Luo X, editors. Multi-user Searchable Encryption with Coarser-Grained Access Control without Key Sharing. International Conference on Cloud Computing and Big Data; 2014: IEEE.
23. Kaci A, Bouabanatebibel T, editors. Access control reinforcement over searchable encryption. IEEE International Conference on Information Reuse and Integration; 2014: IEEE.
24. Lv Z, Chi J, Zhang M, Feng D, editors. Efficiently Attribute-Based Access Control for Mobile Cloud Storage System. IEEE International Conference on Trust, Security and Privacy in Computing and Communications; 2014: IEEE.
25. Shojafar M, Cordeschi N, Baccarelli E. Energy-efficient Adaptive Resource Management for Real-time Vehicular Cloud Services. IEEE Transactions on Cloud Computing. 2016; PP(99):1–14.
26. Li W, Song H. ART: An Attack-Resistant Trust Management Scheme for Securing Vehicular Ad Hoc Networks. IEEE Transactions on Intelligent Transportation Systems. 2016; 17(4):960–9.
27. Umar MM, Mehmood A, Song H. SeCRoP: secure cluster head centered multi-hop routing protocol for mobile ad hoc networks. Security & Communication Networks. 2016.
28. Butun I, Erol-Kantarci M, Kantarci B, Song H. Cloud-centric multi-level authentication as a service for secure public safety device networks. IEEE Communications Magazine. 2016; 54(4):47–53.
29. Xu Q, Ren P, Song H, Du Q. Security Enhancement for IoT Communications Exposed to Eavesdroppers With Uncertain Locations. IEEE Access. 2016; 4:1–12.
30. Shojafar M, Abawajy JH, Delkhah Z, Ahmadi A, Pooranian Z, Abraham A. An Efficient and Distributed file search in Unstructured Peer-to-Peer Networks. Peer-to-Peer Networking and Applications. 2015; 8 (1):120–36.
31. Javanmardi S, Shojafar M, Shariatmadari S, Ahrabi SS. FRTRUST: a fuzzy reputation based model for trust management in semantic P2P grids. International Journal of Grid & Utility Computing. 2014; 6 (1):57–66.
32. Wei W, Fan X, Song H, Fan X. Imperfect Information Dynamic Stackelberg Game Based Resource Allocation Using Hidden Markov for Cloud Computing. IEEE Transactions on Services Computing. 2016:1–13.

33. Zhang Y, Sun L, Song H, Cao X. Ubiquitous WSN for Healthcare: Recent Advances and Future Prospects. IEEE Internet of Things Journal. 2014; 1(1):311–8.

34. Kallahalla M, Riedel E, Swaminathan R, Wang Q, Fu K, editors. Plutus: Scalable Secure File Sharing on Untrusted Storage. Usenix Conference on File and Storage Technologies; 2003: USENIX association; 2003.

35. Goyal V, Jain A, Pandey O, Sahai A. Bounded Ciphertext Policy Attribute Based Encryption: Automata, Languages and Programming; 2015. 579–91 p.

36. Dan B, Crescenzo GD, Ostrovsky R, Persiano G. Public Key Encryption with Keyword Search: Springer Berlin Heidelberg; 2004. 506–22 p.

37. Duquesne S, Lange T. Pairing-based cryptography. Mathiiscernetin. 2004; volume 22(3):573–90.