PLOS ONE

RESEARCH ARTICLE

Searchable and revocable multi-data owner attribute-based encryption scheme with hidden policy in cloud storage

Shangping Wang[1], **Tingting Gao**[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

* gtting2010@126.com

Citation: Wang S, Gao T, Zhang Y (2018) Searchable and revocable multi-data owner attribute-based encryption scheme with hidden policy in cloud storage. PLoS ONE 13(11): e0206126. https://doi.org/10.1371/journal.pone.0206126

Editor: Feng Lin, University of Colorado Denver, UNITED STATES

Received: March 14, 2018

Accepted: October 8, 2018

Published: November 1, 2018

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, http://www.nsfc.gov.cn) to SW, the Key Project of Research Foundation of Natural Science Foundation of Shaanxi Province of China (NO. 2016JZ001, http://www.sninfo.gov.cn/) to SW. The funders had no role in study design, data collection

Abstract

With the development of outsourcing data services, data security has become an urgent problem that needs to be solved. Attribute-based encryption is a valid solution to data security in cloud storage. There is no existing scheme that can guarantee the privacy of access structures and achieve attribute-based encryption with keyword search and attribute revocation. In this article, we propose a new searchable and revocable multi-data owner attribute-based encryption scheme with a hidden policy in cloud storage. In the new scheme, the same access policy is used in both the keyword index and message encryption. The advantage of keyword index with access policy is that as long as a user's attributes satisfy the access policy, the searched ciphertext can be correctly decrypted. This property improves the accuracy of the search results. The hidden policy is used in both the ciphertext and the keyword index to protect users' privacy. The new scheme contains attribute revocation, which is suitable for the actual situation that a user's attributes maybe changed over time. In the general bilinear group model, the security of the scheme is demonstrated, and the efficiency of the scheme is analyzed.

1. Introduction

With technological developments, enterprise and personal data, photos, documents, and even health records maybe outsourced to cloud storage. Jiang D et al. [1] proposed a way to solve the network routing problem in cloud computing, it can achieve higher network energy efficiency for cloud computing. Siddiqui Z et al. [2] proposed in the dynamic cloud environment, the application of telemedicine information system provides convenience for patients and doctors. Along with the many benefits that cloud storage provides, it also presents serious data security problems. The data uploaded to the cloud should be encrypted to prevent information leakage. However, traditional encryption methods cannot be used to achieve access control and keyword searches. Therefore, we ask the following question. How can the data owners encrypt their data and enable both access control and quick searching in cloud storage?

Waqar A et al. [3] proposed a framework for preservation of cloud users' data privacy using dynamic reconstruction of metadata, it can protect the cloud users' data privacy. Lin H Y et al.

and analysis, decision to publish, or preparation of the manuscript.

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

[4] proposed a scheme by use of threshold encryption and group signature mechanism to ensure the security of transmission data, it can ensure that the split and merged messages are not broken.

In traditional public-key cryptography, a message is encrypted for a specific receiver using the receiver's public-key. Identity-based cryptography and in particular identity-based encryption (IBE) changed the traditional understanding of public-key cryptography by allowing the public-key to be an arbitrary string, e.g., the email address of the receiver. ABE goes one step further and defines the identity not atomic but as a set of attributes, e.g., roles, and messages can be encrypted with respect to subsets of attributes (key-policy ABE—KP-ABE) or policies defined over a set of attributes (ciphertext-policy ABE—CP-ABE). The key issue is, that someone should only be able to decrypt a ciphertext if the person holds a key for "matching attributes" (more below) where user keys are always issued by some trusted party. Attribute-based encryption technology can not only protect the privacy of data, but also solve the problem of information sharing in practical application. For attribute-based encryption scheme, data access control is an effective way to ensure data security. Attribute-based encryption enables fine-grained access control for data. A security issue in the cloud environment is the search problem. The data in cloud servers is stored in ciphertext, which guarantees the privacy of data. Once a user needs to find a relevant document containing a keyword, he will encounter the problem of how to search. The server performs a search operation, but does not know what the user is searching for. It can effectively protect the privacy of user search. Of course, in a cloud storage system, data access is not static. For example, if an employee is fired or promoted, the corresponding attribute needs to be changed. The attribute encryption technology supports multiple data owners to upload encrypted personal information records, and can conduct multiple keyword searches. It also allows data owners to search for different periods of time for multiple users.

In the existing attribute-based encryption schemes, the cloud server must know the accessing strategy to perform the keyword search operation. This requirement makes it a difficult task to simultaneously achieve searchability and protect the privacy of the access control. The hidden strategy can protect the privacy of the user's attributes, and the user's attributes may frequently change in practice. Therefore, the attribute revocation mechanism is essential. An attribute change for a single user may lead to changes of other users' private keys that are associated with the attribute and even the changes of the ciphertext corresponding to the attribute.

How to structure a searchable and revocable attribute-based encryption scheme with hidden policy for multi-data owners in cloud storage is a challenging problem.

1.1 Advantages of the scheme

In this scheme, we proposed a searchable and revocable attribute-based encryption scheme with hidden policy for multi-data owners in cloud storage. The primary advantages of the scheme are summed up as follows:

- In our new scheme, the same access policy is used in message encryption and keyword index construction. The benefit of using the access policy in the construction of the keyword index is that as long as a user's attributes satisfy the access structure, when the user submits a search token containing the secret attribute key to the search server, the search server can search the documents the user is interested in. The search results can then be decrypted by the user. Thus, the access policy is considered in the search process, which improves the accuracy of the search results.
- The access policy is hidden in the ciphertext and keyword index. The hidden policy can protect the privacy of the user's attributes.

- This scheme has the function of attribute revocation. If a user's attribute changes, the index, ciphertext and private key connected with the attribute can be updated in time to ensure the security of the information.
- A search server is introduced in the system. It is used to store the keyword index. The ciphertext is stored in the cloud storage server. A keyword search is performed by the search server. For an authenticated user, the user gives the corresponding search token to a search server, and a search server responds to the search. When his attributes satisfy the access control structure and the given keyword is matched, a search server notifies the cloud storage server to send the relevant ciphertext to the user for decryption.
- In the general bilinear group model, it is demonstrated that the keyword index is secure under the keyword guessing attack and that the ciphertext is indistinguishable under the chosen-plaintext attack.

1.2 Related research

Attribute-based encryption (*ABE*). Sahai and Waters [5] devised the first attribute-based encryption system, which was a historic breakthrough. Subsequently, Bethencourt, Sahai and Waters [6] in 2005 constructed an attribute encryption with ciphertext policy. The model can achieve fine-grained access control of ciphertext through attributes. There are two forms in the *ABE*: the ciphertext policy (*CP–ABE*) and the key policy (*KP–ABE)*. In the ciphertext policy, access control policy is embedded in the encrypted ciphertext, and the private key is related to attribute set. Only when the attributes of the user meet the ciphertext policy can the ciphertext can be decrypted. In the key policy, the ciphertext is related to the description of the attribute set, and the user's private key is related to the access structure. The access policy is defined on the attribute set. When the attribute sets meet the access control, the private key of the attributes can decrypt the corresponding ciphertext. Ling and Newport [7] proposed a provable secure ciphertext policy ABE, but the access structure in their scheme can only support the gate condition. However, none of these schemes supports keyword search ([8, 9, 10]).

Attribute-based encryption with keyword search (*ABKS*). Song et al. [11] proposed the first initial searchable encryption program. In addition to the search results, the server knew nothing about the search keyword. Miao Y et al.[12] proposed a secure cryptographic primitive called as attribute-based multi-keyword search over encrypted personal health records in multi-owner setting to support both fine-grained access control and multi-keyword search via Ciphertext-Policy Attribute-Based Encryption. In [13], the authors propose a searchable encryption scheme for keywords in the hidden strategy. If the data user's attributes do not meet the access policy, the user cannot obtain the information of the access policy and cannot search for the encrypted data. Its innovation lies in the construction of the keyword index for the concealed access structure. However, there is no attribute-based encryption, and there is only a single data owner in the scheme in this article, while in practical situations, there should be multi-data owners in the system. Zheng and Sun ([14, 15]) in 2014 proposed two attribute-based keyword search schemes (*ABKS*). A data owner grants the search ability to users through the setup of an access policy, which effectively improves the search efficiency. The cloud server sends corresponding find results to the user when the user's attributes meet the access control structure, which is specified by data owner. Tang Y et al. [16] constructed a multi-keyword search scheme that applied to the network environment, based on privacy protection and efficiency. Xia Z [17] proposed multiple keyword searches and dynamic updates in cloud storage. Zhong H et al.[18] proposed a decentralized multi-authority CP-ABE access control scheme supporting the user revocation. Guo C et al. [19] constructed the access control of individual

cases stored in the cloud server, it can achieve fine-grained access control for EHR. Also it allows multiple users to search on different databases. Fan Y et al. [20] constructed a verifiable scheme to support multi-keyword search. Guo Z [21] proposed a multi-keyword sorting search and supported the sharing of search functions. However, none of these existing schemes can hide access structures [22].

Attribute-based encryption hidden policy. Lai J et al. [23–26] provided some attribute-based hidden policy encryption schemes. In these schemes, access structure embedded in the ciphertext, for those attributes do not meet the access structure users cannot decrypt the ciphertext.

Attribute-based encryption revocation (ABER). Tian et al. [27] proposed a revocable attribute-based encryption project. In this project, once a user is revoked, it is necessary to assign new keys to all other users in the system, except the revoked user. Then, a new encryption key re-encrypts the ciphertext. Thus, the revoked user can no longer decrypt the ciphertext. The method is not smart, because in practice the revoked users are only a small part and the majority of users are not revoked. Therefore, the method is infeasible. Zhihua Xia[28] presented an attribute-based access control project with valid revocation in cloud computing. The revocation is implemented using the version number of the private key, and the scheme also supports the backward security and forward security. The scheme is demonstrated to be effective and secure. Chen J [29] proposed an attribute-based encryption scheme with revocation and update in the cloud storage, in which the user is directly revoked. Li X et al. [30] proposed the revocation of two factors that are based on attribute encryption under the cloud storage, combining identity and attribute. Liu Z et al. [31] proposed a solution to update the user's access rights in a timely manner once the user's attributes change, and the data owner updates access control. In addition, there are some other articles that discuss the methods of attribute revocation for attribute-based encryption scheme ([32, 33, 34, 35, 36]).

2. Preliminaries

2.1 Bilinear map

Let $(e, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T) \leftarrow BGenMap(1^\lambda)$ be represented by a symmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where λ is a security parameter, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are three multiplicative cyclic groups with the same order of prime p, and $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ are the generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively. The bilinear e meets the following four conditions:

1. Bilinearity: $\forall(g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2, \forall a, b \xleftarrow{R} \mathbb{Z}_p : e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
2. Non-degeneracy: $e(g_1,g_2) \neq 1$;
3. Efficiency: There is a valid polynomial time algorithm to compute $e(g_1,g_2)$, $\forall(g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$;
4. There is a valid, publicly calculated (no need reversible) isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ such that $\psi(g_2) = g_1$.

2.2 Generic bilinear group model

Let $(e, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T) \leftarrow BGenMap(1^\lambda)$ be defined as follows. In a general linear group model, three random codes are assumed as $\varsigma_1, \varsigma_2, \varsigma_T : \mathbb{Z}_p^+ \rightarrow \{0,1\}^m$. $\mathbb{Z}_p^+$ is an addition group, and $m > 3\log p$. For $i = 1,2,T$, let $\mathbb{G}_i = \{\varsigma_i(x) | x \in \mathbb{Z}_p^+\}$. There are three oracles to compute the operation in the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. There are oracles to compute non-degenerate linear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

2.3 Access structure

Definition 1. Index n attributes in the system can be denoted as $U = \{1,2,\cdots,n\}$. For each attribute $i \in U$,let $S_i = \{v_{i,1}, v_{i,2} \cdots v_{i,n_i}\}$ denote all the possible values for this attribute, where n_i is the number of possible values for this attribute i. Suppose that $L = \{L_1,L_2 \cdots L_n\}$ is a list of a user's attributes, where $L_i \in S_i$. $P = \{P_1,P_2 \cdots P_n\}$ is an access structure, where $P_i \subseteq S_i$. Define $L| = P$ if the attribute list $L = \{L_1,L_2 \cdots L_n\}$ meets the access structure $P = \{P_1,P_2 \cdots P_n\}$. In other words $L_i \in S_i$, for all i,$1 \leq i \leq n$.

3. System model and security model

3.1 System entities

Above all, the system framework is described in Fig 1.The framework includes the five main entities of the trusted authority, the cloud storage server, the search server, the multi-data owners, and multi-users. Particularly, the trusted authority controls the common parameters and distributes certified users' private keys. The private key is related with the user's attribute list. The cloud storage server provides storage capabilities. Data owners encrypt messages, construct keyword indexes. Owners outsource encrypted messages to the cloud. The keyword index is outsourced to the search server, and the search server is responsible for matching. A certified data user in the system can generate a keyword search token related to his attributes' private key. A search token is presented to a search server, and the search server searches for the keyword index. If the user's attribute list meets the access structure implied in the keyword

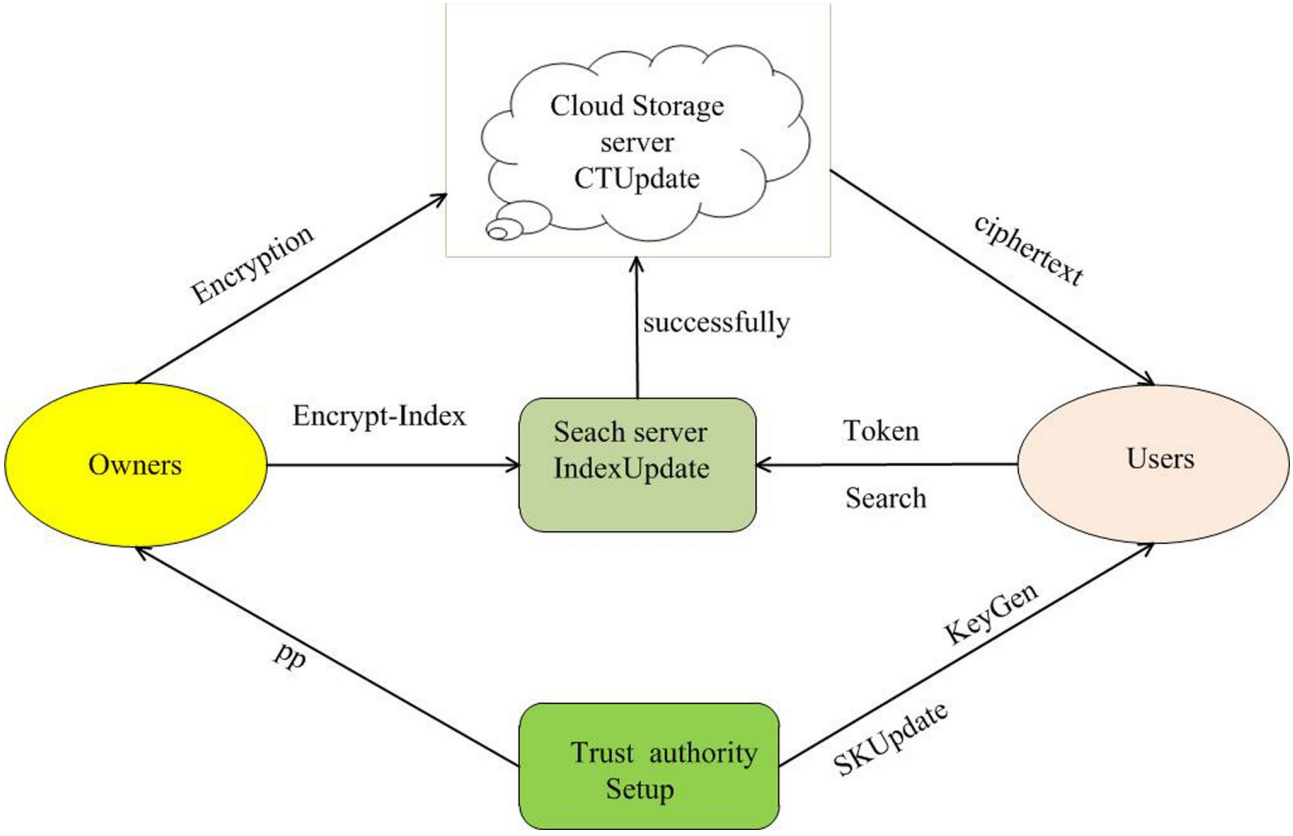

Fig 1. System model.

https://doi.org/10.1371/journal.pone.0206126.g001

index, the successful search is returned to a cloud server. Next, the cloud storage server sends the ciphertext corresponding to the keyword index to the user.

3.2 Function definition

Definition 2: Our scheme contains a set of polynomial time algorithms

$$\Pi = \begin{pmatrix} Setup, KeyGen, Encrypt, Encrypt - Index, GenToken, \\ Search, Decrypt, SKUpdate, CTUpdate, IndexUpdate \end{pmatrix} \quad (1)$$

as described below.

Setup $(1^{\lambda})\rightarrow(msk,pp)$: The algorithm is executed by a trusted authentication attribute authority. It takes the security parameter λ as an input and outputs the public parameter pp and the master key msk.

KeyGen $(msk,pp,L)\rightarrow sk$: The algorithm is used to produce a user's private key by a trusted authority. It takes the master key msk, the public parameter pp,and the users attribute list L as the inputs. It outputs the key sk associated with L.

Encryption $(pp,m,P)\rightarrow ct$: A data owner executes the algorithm. It takes the common parameters pp,a message m, and one access control p as inputs. It outputs a ciphertex ct.

Encrypt-Index $(pp,w,P)\rightarrow Index$: A data owner executes the algorithm. It takes the common parameters pp, a set of keywords w, and an access control structure p as inputs. It then exports the key $Index$.

GenToken $(sk,w)\rightarrow tok$: A data user executes the algorithm to produce search tokens for queries. It takes the input private key sk and a keyword w as inputs. It then exports the keyword search token tok.

Search $(tok,Index)\rightarrow\{0,1\}$: A search server runs the algorithm. It takes the keyword index $Index\leftarrow(pp,w,P)$ and a search token $tok\leftarrow(sk,w')$ as inputs. It then outputs 1 if $L\mapsto P$ and $w = w'$. Otherwise, it outputs 0.

Decryption $(CT,sk)\rightarrow m$: A data user executes the decryption algorithm. It takes the ciphertext ct and the decryption key sk as inputs and outputs message m.

CTUpdate $(\hat{c}_{i,j,2}, u_{i,j}) \rightarrow \hat{c}'_{i,j,2}$, The cloud storage server executes the ciphertext update algorithm. It takes ciphertext ct and update operator $(v_{i,j},u_{i,j})$ as inputs. It then outputs the updated ciphertext ct'.

SKUpdate $(K_{i,t_i,1}, u_{i,j}) \rightarrow K'_{i,t_i,1}$: The authority executes the user's private key update algorithm. It takes the user's private key $K_{i,t_i,1}$ and the update operator $(v_{i,j},u_{i,j})$ as inputs. It then outputs the updated key sk'.

IndexUpdate $(I_{i,j,2}, u_{i,j}) \rightarrow I'_{i,j,2}$:The search server executes the update index algorithm. It takes the update operator $(v_{i,j},u_{i,j})$as an input and then exports the updated index $Index'$.

3.3 Security definition

1.The secure game of indistinguishability of keyword index under the selective keyword attack with hidden policy.

System establishment: The adversary selects two access control strategies P_0,P_1.He needs to send them to the challenger. The challenger selects safety parameters λ and runs the Setup(λ) algorithm, generating public parameters pp and mask secret key msk.The challenger gives the public parameter pp to the adversary and leaves the master secret key msk.

Phase 1. The adversary selects a list of attributes L such that $L| \neq P_0 \wedge L| \neq P_1$. He then asks in polynomials times as follows:

$O_{KeyGen}(L)$. The challenger generates the private key sk though KeyGen $(msk,pp,L)\rightarrow sk$ and gives it to the adversary.

$O_{GenToken}(L,w)$. The challenger generates sk through $O_{KenGen}(L)$. He then runs the token-generating algorithm GenToken $(sk,w)\rightarrow tok$ to get the token and return it to the adversary.

Challenge. The adversary submits two challenge keywords w_0,w_1 to the challenger. The condition is that the adversary has not asked for any search tokens of w_0,w_1. The challenger chooses a random bit $b\in\{0,1\}$. He then produces the index I_b by the index generating algorithm for keyword w_b under policy P_b, and returns the index I_b to the adversary $\mathcal{A}$.

Phase 2. The adversary $\mathcal{A}$ can still query similar to Phase 1.The restricted condition is $w \neq w_0,w_1$.

Guess: The adversary $\mathcal{A}$ outputs a guess b' for b. If $b' = b$, then the adversary wins this game.

The adversary's advantage in the game is defined as

$$Adv(\mathcal{A}) = |\Pr[b' = b] - 1/2| \tag{2}$$

If there is no polynomial time, the adversary can win the above game with a non-negligible advantage. Next, the scheme is called secure in the sense of the indistinguishability of keyword index under the selective keyword attack with the hidden policy.

2. The secure game of indistinguishability of ciphertext under the selective plain-text attack with the hidden policy.

System establishment. The adversary selects two access control strategies P_0,P_1. He then transmits them to a challenger. The challenger selects the safety parameter λ and runs the Setup(λ) algorithm, generating public parameters pp and master secret key msk.The challenger gives the public parameter pp to the adversary and later leaves the master secret key msk.

Phase 1. The adversary selects a list of attributes L such that $L| \neq P_0 \wedge L| \neq P_1$ and asks in polynomials times as follows $O_{KeyGen}(L)$.The challenger generates the private key sk though KeyGen $(msk,pp,L)\rightarrow sk$ and provides it to the adversary.

Challenge. The adversary submits two equal length messages m_0,m_1 to the challenger. The challenger selects a random bit $b\in\{0,1\}$. He subsequently generates the ciphertext ct_b by the encryption algorithm for message m_b under policy P_b and returns the ciphertext ct_b to the adversary $\mathcal{A}$.

Phase 2. The adversary $\mathcal{A}$ can still query similar as Phase 1.

Guess. The adversary $\mathcal{A}$ exports a guess b' for b. If $b' = b$, then he wins this game. The adversary's advantage in the game is defined as

$$Adv(\mathcal{A}) = |\Pr[b' = b] - 1/2| \tag{3}$$

If there is no polynomial time adversary that can win the above game with a non-negligible advantage, then the scheme is called the secure model of indistinguishability of ciphertext under the selective plain-text attack with hidden policy.

4. Scheme construction

In this part, we will propose a searchable and revocable attribute-based encryption scheme with a hidden policy in cloud storage. Our new scheme achieves encryption and keyword search and attribute revocation. The access control policy consists of a set of *AND* gates. In the system we assume that there are n attributes, and all the attributes are labeled as $\{1,2,\cdots n\}$.

4.1 Motivation

Authors in [13] proposed a searchable encryption scheme with the hidden strategy, and the hidden strategy is a major feature of the scheme. However, we find that the scheme is correct only

if the data owner and the data user are the same one, this error is not easy to find as they use the same parameter r for the data owner and data user. Here we will give a detail analysis, in order to demonstrate this problem, we use different parameter r for the data owner and data user.

The data owner's parameters can be denoted by $X_0 = Y^{x_o}, C_{u_0} = X_0^{-r_0}(x_0 \in \mathbb{Z}_p), \tilde{C}_0 = Y^{r_0}$. in which r_0 is randomly selected by the data owner, x_0 is randomly selected by attribute authority for the data owner. Data user's parameters denoted by $X_u = Y^{x_u}, C_{u_u} = X_u^{-r_u}(x_u \in \mathbb{Z}_p)$, $\tilde{T} = x_u + s$,in which r_u is randomly selected by the data user, x_u is randomly selected by attribute authority for the data user. In the search algorithm, the match equation should be $\tilde{C}^{\tilde{T}} \cdot C_{u_u} = Y^{r_0 s} = e(g_1, g_2)^{r_0 s\alpha}$, but if the data owner and the data user are not the same one, we find that $\tilde{C}^{\tilde{T}} \cdot C_{u_u} = \tilde{C}^{(x_u+s)} \cdot X_u^{-r_u} = Y^{r_0(x_u+s)} \cdot Y^{-x_u r_u} = Y^{x_u(r_0-r_u)} \cdot Y^{r_0 s}$, which is not equal to $Y^{r_0 s}$ except for $r_0 = r_u$. Thus the original scheme is correct only for the data owner and the data user are same one. This limits the usefulness of the scheme in [13].

To overcome these problems, we improve the scheme in reference [13].We will take the public parameter r and design a new scheme that accounts for multiple data owners. Another improvement of our new scheme is to simplify the hash function with a key [13] to a general hash function without a key to increase the practicality of the scheme. Since all users in the system share a secret key for the hash function, it is actually not secure, and the server can easily collude with a user to get the key. We also add attribute encryption and attribute revocation to make the scheme feasible and retain the advantages of the hidden access structure.

4.2 Our construction

The scheme consists of the following algorithms.

- *Setup*(1^λ): Input the security parameter λ. Then, the algorithm produces the public parameters and the master secret key as follows:

  1. *Generate*$(1^\lambda) \rightarrow (e, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, H)$, where e is a symmetric bilinear mapping e:$\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. g_1,g_2 are the generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively. $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are three multiplicative cyclic groups of prime orders p. $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ is a secure hash function.

  2. For each attribute i,$1 \le i \le n$. Let $S_i = \{v_{i,1}, v_{i,2} \cdots v_{i,n_i}\}$ be a set of all possible values of attribute i. It generates a random values set $\{a_{i,j} \in \mathbb{Z}_p\}_{1 \le j \le n_i}$ for attribute i and calculates $\{A_{i,j} = g_1^{a_{i,j}}\}_{1 \le j \le n_i}$.It chooses $\alpha, \beta, b \xleftarrow{R} \mathbb{Z}_p$, and calculates $Y = e(g_1, g_2)^\alpha, B = g_1^b$, and $K_0 = g_2^{\frac{\alpha+\beta}{b}}$. It chooses a random number $r \xleftarrow{R} \mathbb{Z}_p$ and publishes it and then calculates $\hat{I} = Y^r, I_0 = B^r$. The public parameter pp and the master secret key msk are set as follows:

$$pp = (e, p, g_1, g_2, g_2^\beta, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, H, Y, B, K_0, \{\{A_{i,j}\}_{1 \le j \le n_i}\}_{1 \le i \le n}, r, \hat{I}, I_0) \quad (4)$$

$$msk = (\alpha, \beta, b, \{\{a_{i,j}\}_{1 \le j \le n_i}\}_{1 \le i \le n}). \quad (5)$$

- *KeyGen*(msk,pp,L):Suppose $L = \{L_1, L_2, \cdots L_n\} = \{v_{1,t_1}, v_{2,t_2}, \cdots v_{n,t_n}\}$ is the attribute list of a user U. User U chooses one of his own $x_u \xleftarrow{R} \mathbb{Z}_p$, calculates $X_u = Y^{x_u}$, and then submits it to the attribute authority. The user saves x_u.Next, for each attribute i,$1 \le i \le n$, the authority chooses $\lambda_i \xleftarrow{R} \mathbb{Z}_p$ and calculates $K_{i,t_{i,1}} = g_2^{\beta + a_{i,t_i}\lambda_i}, K_{i,2} = g_2^{\lambda_i}$. It finally sets the private key of

the user with the attribute list $L = \{L_1, L_2, \cdots L_n\} = \{v_{1,t_1}, v_{2,t_2}, \cdots v_{n,t_n}\}$ as follows:

$$sk = (x_u, \{K_{i,t_i,1}, K_{i,2}\}_{1 \le i \le n}). \quad (6)$$

The authority add tuples (U,I_u,L) to the list of users U_{List}, where $I_u = X_u^{-r}$. Next, the authority sends U_{List} to the search server.

- *Encrypt*(*pp*,*m*,*P*):Suppose $P = \{P_1,P_2,\cdots P_n\}$ is an access control policy, $P_i \subseteq S_i$. When outsourcing a file F to a cloud storage server, the algorithm produces a ciphertext that is related to the access control structure P as follows:

  1. Above all randomly chooses $\hat{r} \xleftarrow{R} \mathbb{Z}_p$, $\hat{c} = me(g_1, g_2)^{\alpha\hat{r}}$, and $\hat{c}_0 = B\hat{r}$,where m is the key to encrypt the file F by a symmetric encryption algorithm.
  2. For each i,$1 \le i \le n$, it first selects $\hat{r}_i \xleftarrow{R} \mathbb{Z}_p$ such that $\hat{r} = \sum_{i=1}^{n} \hat{r}_i$ and it calculates $\hat{c}_{i,1} = g_1^{\hat{r}_i}$. For all $v_{i,j} \in S_i$, if $v_{i,j} \in P_i$, set $\hat{c}_{i,j,2} = A_{i,j}^{\hat{r}_i}$.If $v_{i,j} \notin P_i$, it sets $\hat{c}_{i,j,2}$ to a random value within $\mathbb{G}_1$. It sets the ciphertext as

$$CT = (\hat{c}, \hat{c}_0, \{\hat{c}_{i,1}, \{\hat{c}_{i,j,2}\}_{1 \le j \le n_i}\}_{1 \le i \le n}). \quad (7)$$

- *Encrypt–Index*(*pp*,*w*,*P*):Suppose $P = \{P_1,P_2,\cdots P_n\}$ is an access control policy, which is the same access control policy as it is in the encrypted ciphertext. Let w be a keyword extracted from file F. It produces a secure keyword index related to the access control policy P as follows:

For each attribute i,$1 \le i \le n$, above all selects $r_i \xleftarrow{R} \mathbb{Z}_p$, makes $r = \sum_{i=1}^{n} r_i$, and computes $I_{i,1} = g_1^{r_i}$. For all $v_{i,j} \in S_i$, if $v_{i,j} \in P_i$, $I_{i,j,2} = A_{i,j}^{r_i/H(w)}$ is set. If $v_{i,j} \notin P_i$, $I_{i,j,2}$ is set to a random value in $\mathbb{G}_1$. The keyword index of the keyword w is

$$Index = (\{I_{i,1}, \{I_{i,j,2}\}_{1 \le j \le n_i}\}_{1 \le i \le n}). \quad (8)$$

If the file has multiple keywords, it can be used to generate multiple security indexes.

Note that here r is the public system parameter and is the same for all keyword index generations.

- *GenToken*(sk, $\tilde{w}$): This algorithm generates a secure search token for a keyword $\tilde{w}$. If a user U wants to search a keyword $\tilde{w}$,the user U chooses a $s \xleftarrow{R} \mathbb{Z}_p$ and then computes $t = x_u + s$, $T_0 = K_0^s$ for each i,$1 \le i \le n$. It computes $T_{i,t_i,1} = K_{i,t_i,1}^s$, $T_{i,2} = K_{i,2}^{H(\tilde{w})s}$. Finally, it sets the search Token as

$$tok = (t, T_0, \{T_{i,t_i,1}, T_{i,2}\}_{1 \le i \le n}, U). \quad (9)$$

- *Search*(*tok*,*Index*): Once the token of user U is received, the search server first checks whether the U is in U_{List}. If it is not, the request s refused. Otherwise, it gets the tuples (U,I_u, L),where $L = \{L_1, L_2, \cdots L_n\} = \{v_{1,t_1}, v_{2,t_2}, \cdots v_{n,t_n}\}$ and $I_u = X_u^{-r}$.The search server runs the matching algorithm for (*tok*,*Index*) and (U,I_u,L) as follows:

  1. It computes $E_1 = \prod_{i=1}^{n} e(I_{i,1}, T_{i,t_i,1})$.
  2. For each i,$1 \le i \le n$, if $L_i = v_{i,t_i}$, it chooses $I_{i,t_i,2}$ and computes $E_2 = \prod_{i=1}^{n} e(I_{i,t_i,2}, T_{i,2})$ and $E = E_1/E_2 = e(g_1,g_2)^{srβ}$. If $e(I_0, T_0) \cdot E^{-1} = \hat{I}^t \cdot I_u$, the match is successful and returns 1.A

notification is sent to the cloud server. The cloud server sends the corresponding ciphertext associated with the index to the user. Otherwise, the match is failed and returns 0.

- *Decrypt*(*CT*,*sk*): The decryption algorithm is run by data user U with attribute list $L = \{L_1, L_2, \cdots L_n\} = \{v_{1,t_1}, v_{2,t_2}, \cdots v_{n,t_n}\}$ to decrypt the ciphertext CT by using its secret key sk. For each i,$1 \leq i \leq n$ and $L_i = v_{i,t_i}$, it chooses $\hat{c}_{i,t_i,2}$ and computes

$$\hat{E} = \prod_{i=1}^{n} e(\hat{c}_{i,1}, K_{i,t_i,1}) / \prod_{i=1}^{n} e(\hat{c}_{i,t_i,2}, K_{i,2}),$$
$$m = \frac{\hat{E}\hat{c}}{e(\hat{c}_o, K_0)}. \tag{10}$$

- *AttriUpdate*($v_{i,j}$,$a_{i,j}$): When a user's attribute i is revoked, suppose that the attribute value revoked is $v_{i,j}$.The authority runs this update algorithm and selects a new random value $a'_{i,j}$ instead of the old secret value $a_{i,j}$ corresponding to $v_{i,j}$. It publishes an attribute update operator ($v_{i,j}$,$u_{i,j}$), where $u_{i,j} = a'_{i,j}/a_{i,j} \bmod p$.

- *PPUpdate*($A_{i,j}$,$u_{i,j}$): The authority inputs the update operator ($v_{i,j}$,$u_{i,j}$) and recalculates $A'_{i,j} = A_{i,j}^{u_{i,j}}$ as the new system parameter to instead of the old parameter $A_{i,j}$ for attribute i and publishes it in the system parameter set.

- *CTUpdate*($\hat{c}_{i,j,2}$, $u_{i,j}$): The ciphertext update algorithm inputs the ciphertext $\hat{c}_{i,j,2}$ and the update operator ($v_{i,j}$,$u_{i,j}$). It then outputs the new ciphertext $\hat{c}_{i,j,2}$ such that

$$\hat{c}'_{i,j,2} = (\hat{c}_{i,j,2})^{u_{i,j}} = g_1^{r_i a'_{i,j}}. \tag{11}$$

- *SKUpdate*($K_{i,1}$,$u_{i,j}$): The private key update algorithm inputs the private key $K_{i,1}$ corresponding to attribute $v_{i,j}$ and the update operator ($v_{i,j}$,$u_{i,j}$). It then outputs the new private key $K'_{i,t_i,1}$ as

$$K'_{i,t_i,1} = (K_{i,t_i,1}/g_2^{\beta})^{\lambda_i u_{i,j}} \cdot g_2^{\beta} = g_2^{\beta + \lambda_i a_{i,t'_i}}. \tag{12}$$

- *IndexUpdate*($I_{i,j,2}$,$u_{i,j}$): The index update algorithm inputs $I_{i,j,2}$ (which is part of the index related to attribute $v_{i,j}$) and the update operator ($v_{i,j}$,$u_{i,j}$). It then outputs the new index $I'_{i,j,2}$ as

$$I'_{i,j,2} = (I_{i,j,2})^{u_{i,j}} = (g_1^{a_{i,j} r_i / H(w)})^{a'_{i,j}/a_{i,j}} = g_1^{r_i a'_{i,j}/H(w)}. \tag{13}$$

5. Security analysis

The correctness of algorithm *Search*(*tok*,*Index*) is as follows:

If $L| = P$, $w = w'$, and $L = \{L_1, L_2, \cdots L_n\} = \{v_{1,t_1}, v_{2,t_2}, \cdots v_{n,t_n}\}$,

$$E = \frac{E_1}{E_2} = \frac{\prod_{i=1}^{n} e(I_{i,1}, T_{i,t_i,1})}{\prod_{i=1}^{n} e(I_{i,t_i,2}, T_{i,2})} = \frac{\prod_{i=1}^{n} e\left(g_1^{r_i}, g_2^{(\beta + a_{i,t_i}\lambda_i)s}\right)}{\prod_{i=1}^{n} e\left(g_1^{a_{i,j,2} r_i \frac{1}{H(w)}}, g_2^{\lambda_i H(\tilde{w}) s}\right)} = \frac{\prod_{i=1}^{n} e(g_1, g_2)^{sr_i(\beta + a_{i,t_i}\lambda_i)}}{\prod_{i=1}^{n} e(g_1, g_2)^{sr_i \lambda_i a_{i,j,2}}}$$
$$= \prod_{i=1}^{n} e(g_1, g_2)^{sr_i\beta} = e(g_1 \cdot g_2)^{sr\beta},$$

If $e(I_0, T_0) \cdot E^{-1} = e(B^r, K_0^s) \cdot E^{-1} = e\left(g_1^{br}, g_2^{\frac{\alpha+\beta}{b}s}\right) \cdot e(g_1, g_2)^{-sr\beta} = e(g_1, g_2)^{rs\alpha}$ is equal to $\hat{I}^t \cdot I_u = Y^{rt} \cdot X_u^{-r} = e(g_1, g_2)^{\alpha r(x_u+s)} \cdot e(g_1, g_2)^{-\alpha x_u r} = e(g_1, g_2)^{rs\alpha}$, the match is successful and the search server returns 1.

The correctness of the decryption algorithm is verified as follows:

If $L| = P$ and $L = \{L_1, L_2, \cdots L_n\} = \{v_{1,t_1}, v_{2,t_2}, \cdots v_{n,t_n}\}$ then

$$\hat{E} = \frac{\prod_{i=1}^{n} e(\hat{c}_{i,1}, K_{i,t_i,1})}{\prod_{i=1}^{n} e(\hat{c}_{i,t_i,2}, K_{i,2})} = \frac{\prod_{i=1}^{n} e\left(g_1^{\hat{r}_i}, g_2^{\beta + a_{i,t_i}\lambda_i}\right)}{\prod_{i=1}^{n} e(g_1^{a_{i,j}\hat{r}_i}, g_2^{\lambda_i})} = \prod_{i=1}^{n} e(g_1, g_2)^{\beta\hat{r}_i} = e(g_1, g_2)^{\beta\hat{r}} \tag{14}$$

$$m = \frac{\hat{E}\hat{c}}{e(\hat{c}_0, K_0)} = \frac{e(g_1, g_2)^{\beta\hat{r}}\ me\ (g_1, g_2)^{\alpha\hat{r}}}{e\left(g_1^{b\hat{r}}, g_2^{\frac{\alpha+\beta}{b}}\right)} \tag{15}$$

Our safety analysis scheme is as follows:

We will analyze and demonstrate the security of our scheme under the general bilinear mapping model ([6, 14, 36]). First, we will prove that our scheme is of the indistinguishability of the keyword index under the selective keyword attack with the hidden policy. Second, we will demonstrate that our scheme is of the indistinguishability of the ciphertext under the selective plain-text attack with the hidden policy.

Theorem 1. Let $\varsigma_1, \varsigma_2, \varsigma_T, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be defined as the general bilinear group model. We request that any adversary $\mathcal{A}$ performs up to q times oracles to ask for group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$'s calculation, including bilinear mapping. In the secure game of the indistinguishability of the keyword index under the selective keyword attack with hidden policy, the advantage of an adversary $\mathcal{A}$ is $O(q^2/p)$.

Proof: Our proof is similar to [13,24].We will design a simulator $\mathcal{B}$ and an adversary $\mathcal{A}$ to perform the indistinguishability of the keyword index under the selective keyword attack with the hidden policy as follows. $\mathcal{A}$ maintains 3 pairs of lists:

$$\begin{aligned} V_{G_1} &= \{\langle F_{1,l}, \varsigma_{1,l}\rangle : l = 1, \cdots, \tau_1\}, \\ V_{G_2} &= \{\langle F_{2,l}, \varsigma_{2,l}\rangle : l = 1, \cdots, \tau_2\}, \\ V_{G_T} &= \{\langle F_{T,l}, \varsigma_{T,l}\rangle : l = 1, \cdots, \tau_T\}. \end{aligned} \tag{16}$$

In these equations, $F_{\tau,l}(\tau\in\{1,2,T\})$ is adversary $\mathcal{A}'$s queries, $\varsigma_{\tau,l}(\tau\in\{1,2,T\})$ is a random string of $\{0,1\}^*$ for each query result, and $\varsigma_{1,l} = \varsigma_1(F_{1,l}), \varsigma_{2,l} = \varsigma_2(F_{2,l}), \varsigma_{T,l} = \varsigma_T(F_{T,l})$.

The initialization definition is $F_{1,1} = 1, F_{2,1} = 1, F_{T,1} = 1$, and $\varsigma_{1,1}, \varsigma_{2,1}, \varsigma_{T,1}$ is the initial mapping string. $\varsigma_1(1)$ represents g_1, $\varsigma_2(1)$ represents g_2, and $\varsigma_T(1)$ represents $e(g_1, g_2)$. In the following query, the adversary $\mathcal{A}$ and the simulator $\mathcal{B}$ use ς to represent the elements in the group. In particular, for each query, the simulator selects random real values contained in the list. Whenever $\mathcal{A}$ gives a query to $\mathcal{B}$, $\mathcal{B}$ will update its list and return to the relevant random string to $\mathcal{A}$. Next, we give $\mathcal{A}$s query as follows:

Group action. Set two operand objects $\varsigma_\tau(x), \varsigma_\tau(y)$. Additionally, $x, y \xleftarrow{R} \mathbb{Z}_p, \tau \in \{1, 2, T\}$. If $\varsigma_\tau(x), \varsigma_\tau(y)$ are not in the list V_{G_T}, they are returned. Otherwise, $\mathcal{B}$ computes $F = x+y \bmod p$ and checks where F is in the list V_{G_T}.If it is in it, it returns $\varsigma_\tau(F)$. Otherwise, $\mathcal{B}$ sets a random string in $\{0,1\}^*$ different from the list V_{G_T} already exists in. Finally, $\mathcal{B}$ will be added $\langle F, \varsigma_\tau(F)\rangle$ to the V_{G_T} and we will have answer $\mathcal{A}$ with the string $\varsigma_\tau(F)$.

Isomorphism. Given a string $\varsigma_2(x)$, if it is not in the list V_{G_2}, it terminates $\perp$. Otherwise, if x already exists in the list V_{G_1}, it returns $\varsigma_1(x)$ to $\mathcal{A}$. If not, $\mathcal{B}$ sets a random string $\varsigma_1(x)$ in $\{0,1\}^*$ that is distinct from any existing list V_{G_1}. Finally, $\mathcal{B}$ adds $\langle x,\varsigma_1(x)\rangle$ to V_{G_1}, and sets $\varsigma_1(x)$. It then returns to $\mathcal{A}$.

Bilinear pairing. Given two operations $\varsigma_1(x),\varsigma_2(y)$, if $\varsigma_1(x)$ not in the list V_{G_1} and $\varsigma_2(y)$ not in the list V_{G_2}, it terminates $\perp$. Otherwise, $\mathcal{B}$ calculates $F = xy \bmod p$ also checks if F in the list V_{G_T}. In that case, $\mathcal{B}$ returns to $\varsigma_T(F)$. Otherwise, $\mathcal{B}$ sets a random $\varsigma_T(F)$ in $\{0,1\}^*$ different from any existing V_{G_T}. Finally, $\mathcal{B}$ will add $\langle F,\varsigma_T(F)\rangle$ to the V_{G_T} and reply $\mathcal{A}$ with string $\varsigma_T(F)$.

Based on the basic operations of the above group, the simulation selects the security game as follows:

Establishment. Adversary $\mathcal{A}$ selects two different challenges with access control policies P_0, P_1. Here, $P_i = \{P_{i,1},P_{i,2},\ldots,P_{i,n}\}$ where $i\in\{0,1\}$, and sends them to $\mathcal{B}$.$\mathcal{B}$ does not select the true value for the variables of the master key $(\alpha, \beta, b, \{\{a_{i,j}\}_{1\le j\le n_i}\}_{1\le i\le n})$, and just maintains it in the corresponding list. Then, $\mathcal{B}$ updates the list by adding a random string representation of tuples $(e(g_1,g_2)^{\alpha}, g_1^{b}, g_2^{\alpha+\beta/b}, e(g_1,g_2)^{\alpha r}, g_1^{br}, r, \{\{g_1^{a_{ij}}\}_{1\le j\le n_i}\}_{1\le i\le n})$ consistent with each common parameter. Finally, $\mathcal{B}$ sets up a new update list to $\mathcal{A}$.

Phase1. $\mathcal{A}$ selects a list of attributes $L = \{L_1, L_2, \cdots, L_n\} = \{v_{1,t_1}, v_{2,t_2}\cdots, v_{n,t_n}\}$. For $O_{KeyGen}(L)$,$O_{GenToken}(L,w)$ queries, the premise is that $\mathcal{A}$ cannot ask the private key and token that satisfies the attributes of the access structure. The process is as follows:

$O_{KeyGen}(L)$:First, $\mathcal{B}$ uses $\langle\alpha,\varsigma_T(\alpha)\rangle$ instead of $e(g_1,g_2)^{\alpha}$. It adds new tuples $\langle\alpha x_n,\varsigma_T(\alpha x_n)\rangle$ of $e(g_1,g_2)^{\alpha x_u}$ by the rules defined above, using variables x_u to the list L_{G_T}.Next, $\mathcal{B}$ increasing tuples $(x_u, \{\{g_2^{\beta+a_{i,t_i}\lambda_i}, g_2^{\lambda_i}\}_{1\le j\le n_i}\}_{1\le i\le n})$ to update the list consistent with the private key, where β,λ_i is the new variable.

$O_{GenToken}(L,w)$:$\mathcal{B}$ runs $O_{KeyGen}(L)$.The list is then updated by adding tuples $(x_u + s, g_2^{(\alpha+\beta)s/b}, \{g_2^{(\beta+a_{i,t_i}\lambda_i)s}, g_2^{\lambda_i H(w)s}\}_{1\le i\le n})$ consistent with the search token. s is the new variable.

Challenge: $\mathcal{A}$ chooses two keywords w_0,w_1. It then inputs $\langle w_0,P_0\rangle$,$\langle w_1,P_1\rangle$ in the real choice of security games. The challenger chooses $\sigma \xleftarrow{R} \{0,1\}$ to encrypt w_σ. Using P_σ, the challenge index ciphertext of $\mathcal{B}$ is as follows: $(\{I_{i,1}, \{I_{i,j,2}\}_{1\le j\le n_i}\}_{1\le i\le n})$.

For $\{I_{i,1}\}_{1\le i\le n}$, $\mathcal{B}$ adds tuples $\langle r_i,\varsigma_1(r_i)\rangle$ to the list V_{G_1}, and the new variable r_i satisfies $r = \sum_{i=1}^{n} r_i$.

For $\{\{I_{i,j,2}\}_{1\le j\le n_i}\}_{1\le i\le n}$,if $w_0 = w_1$ and $v_{i,t_i} \in P_{0,i} \wedge v_{i,t_i} \in P_{1,i}$, $\mathcal{B}$ adds tuples $\langle a_{i,j}r_i/H(w), \varsigma_1(a_{i,j}r_i/H(w))\rangle$ to list V_{G_1}.Otherwise, if $v_{i,t_i}\notin P_{0,i} \wedge v_{i,t_i}\notin P_{1,i}$, $\mathcal{B}$ adds tuples $\langle r_{i,j},\varsigma_1(r_{i,j})\rangle$ to list V_{G_1} with a new variable $r_{i,j}$.If $v_{i,t_i}\notin P_{0,i} \wedge v_{i,t_i} \in P_{1,i}$ or $v_{i,t_i} \in P_{0,i} \wedge v_{i,t_i}\notin P_{1,i}$, $\mathcal{B}$ increases tuples $\langle\theta,\varsigma_1(\theta)\rangle$ with a new variable θ to the list V_{G_1}.

Phase2. $\mathcal{A}$ repeats phase 1 of the inquiry. The requirement is that if $w_0 \neq w_1$, $\mathcal{A}$ cannot ask $O_{KeyGen}(L)$,$O_{GenToken}(L,\omega)$ when $L| = P_0 \wedge L| = P_1$.

After making at most q queries, $\mathcal{A}$ terminates and returns to guessing $\sigma'\in\{0,1\}$. At this point, $\mathcal{B}$ selects a random value of $\sigma \xleftarrow{R} \{0,1\}$ and obtains the real challenge ciphertext. In list V_{G_1}, g_1^{θ} is replaced by $g_1^{a_{i,j}r_i/H(w_\sigma)}$.Finally, $\mathcal{B}$ returns a list of all the updated tuples to $\mathcal{A}$.

Next, a detailed analysis of the $\mathcal{B}$ simulation is presented. The simulation of $\mathcal{B}$ is perfect if and only if no unexpected collisions occur. The so-called collision is for two different polynomials $F_{\tau,l},F_{\tau,l'}(\tau\in\{1,2,T\})$.For some l,l', the corresponding random coding string of the

difference cannot equal 0. Therefore, $F_{\tau,l}-F_{\tau,l'}=0$, and this unexpected collision occurs in the following two conditions.

In front of the replacement, on this occasion, we use theorem [37, 38].The probability of a collision occurring in list V_{G_1}, V_{G_2}, V_{G_T} is expected to be $O(q^2/p)$ at most. For more details, refer to [37, 38].

After the replacement, it is proven that no new equations $F_{k,b}F_{k,l'}$ can be created between polynomials after simulation, even if $\mathcal{B}$ is replaced by $a_{i,j}r_i/H(w_\sigma)$ for θ. We must note that an adversary cannot construct a query for a nonzero $F=F_{k,l}-F_{k,l'}$. It only occurs after substitution when $F=0$. In an alternative security game, the adversary tries to distinguish $g_1^{a_{i,j}r_i/H(w_0)}, g_1^{a_{i,j}r_i/H(w_1)}$ between two different keyword w_0,w_1 queries. Given $\delta_1 \xleftarrow{R} Z_p$, the probability of distinguishing $g_1^{a_{i,j}r_i/H(w_0)}$ and $g_1^{\delta_1}$ is half the probability for adversary $\mathcal{A}$ to distinguish $g_1^{a_{i,j}r_i/H(w_0)}$ *and* $g_1^{a_{i,j}r_i/H(w_1)}$.

As a result, we revise the game in order to determine whether $\mathcal{A}$ would be able to structure the queries of $e(g_1,g_2)^{\gamma a_{i,j}r_i}$ for some g_2^{γ}.Then, it can distinguish $g_1^{a_{i,j}r_i/H(w_0)}$ and $g_1^{\delta_1}$. We prove that $\mathcal{A}$ cannot structure the queries for $e(g_1,g_2)^{\gamma a_{i,j}r_i}$.

To construct $a_{i,j}r_i$, $a_{i,j}r_i$ is from $a_{i,j}r_i/H(w_\sigma)$ according to the simulation. When $\mathcal{B}$ replaces θ with $a_{i,j}r_i/H(w_\sigma)$, since $w_0 \neq w_1$, it cannot obtain the search tokens that satisfy $L|=P_0 \wedge L|=P_1$. Therefore, even if $\mathcal{B}$ submits a true value $a_{i,j}r_i/H(w_\sigma)$ to θ, it cannot eliminate $a_{i,j}r_i$. We then have, as follows.

Fix any $a_{i,j}r_i$ that arises after $\mathcal{B}$s replacement. We make the assumption that $\mathcal{A}$ can construct a query for $e(g_1,g_2)^{\nu}$ where ν is a non-zero polynomial containing θ, which also turns into zero after $\mathcal{B}$ replaces $a_{i,j}r_i$ for θ.

To construct such a ν, $\mathcal{A}$ must cancel $a_{i,j}r_i$ in ν. To our knowledge, there may be a different attribute value $v_{i,j'}(j' \neq j)$ of L_i in the access policy. Adversary $\mathcal{A}$ is able to get the ciphertext $g_1^{a_{i,j'}r_i}$ of $v_{i,j'}(j' \neq j)$. Therefore, adversary $\mathcal{A}$ can obtain $a_{i,j}r_i$ in two ways. One is by pairing $g_1^{a_{i,j'}r_i}, g_2^{\beta+a_{i,j}\lambda_i}$, and the other is by pairing $g_1^{r_i}, g_2^{\beta+a_{i,j}\lambda_i}$.

If adversary $\mathcal{A}$ pairs $g_1^{a_{i,j'}r_i}, g_2^{\beta+a_{i,j}\lambda_i}$, then adversary $\mathcal{A}$ needs to get information about $g_2^{a_{i,j'}\lambda_i}$. However, in the entire simulation, the user knows $g_2^{a_{i,j}\lambda_i}$ and does not know $g_2^{a_{i,j'}\lambda_i}$.Therefore, $g_2^{a_{i,j'}\lambda_i}$ does not exist. In other words, it cannot be paired, so this situation cannot be achieved.

If adversary $\mathcal{A}$ pairs $g_1^{r_i}, g_2^{\beta+a_{i,j}\lambda_i}$, $\mathcal{A}$ will obtain combination $\beta r_i+a_{i,j}r_i\lambda_i$ of the query. Adversary $\mathcal{A}$ wants to know $\rho' a_{i,j}r_i$. A new variable ρ'' is introduced, making $\rho'=\lambda_i\rho''$. First, $\mathcal{A}$ needs to structure $p''\beta r_i$. As previously known, $r=\sum_{r=1}^{n} r_i$.It can be converted into the construct $p''\beta r$. In the entire simulation, the only way to ask $\mathcal{A}$ to construct $p''\beta r$ is to combine T_0, $I_0.g_2^{\frac{\alpha+\beta}{b}s}, g_1^{br}$ are used to get the query of the combination $\alpha rs+\beta rs$. We need βrs, so to eliminate αrs, $\mathcal{A}$ combines $\hat{I}, \tilde{t}$. $e(g_1,g_2)^{\alpha r}$,x_u+s are used to get tuples $\alpha r(x_u+s)$. Next, $\mathcal{A}$ uses $-\alpha r x_u$ to eliminate $\alpha r x_u$ and obtain αrs. Thus $\mathcal{A}$ obtains the required βrs. A new variable ρ''' is introduced to make the $\rho''=\rho'''s$. By asking for $p'''s(\beta(r-\Sigma_{i' \neq i}r_{i'}))$, you can get $a_{i,j}r_i$. However, $\mathcal{A}$ cannot construct such an inquiry. The reasons are as follows.

Since s is randomly selected by the user, adversary $\mathcal{A}$ does not know its value. Hence, adversary $\mathcal{A}$ cannot find a ρ''' to satisfy $\rho''=\rho'''s$. Therefore, ρ''' is not observed. According to the simulation challenge ciphertext, $I_{i,j,2}$ has $v_{i,t_i} \notin P_{b,i} \wedge v_{i,t_i} \in P_{1-b,i}$. In other words, adversary $\mathcal{A}$ cannot obtain the private key sk and search token for search operations. Here, at least one of the r'_i is unknown, according to [6]. Since it is less than $\sum_{i' \neq i} r'_i$, we cannot get the query about $\rho'''s(\beta(r-\Sigma_{i' \neq i}r_{i'}))$.

Then, it is proven that the encrypted message is secure and still operates in a general group model. The above theorem 1 is used to prove that the ciphertext is not distinguishable under the same access policy.

Theorem 2. Under the condition of Theorem 1, the adversary has the advantage over the ciphertext in the scheme as $O(q^2/p)$.

Proof: The establishment of the system and the basic operations of the group are similar to that in the proof of theorem 1. Therefore, they are not repeated in this study. In the system setup, the attacker chooses the policy that will attack P^*.

Using the above same group, we use $\varsigma_1(1)$ for g_1, $\varsigma_2(1)$ for g_2, and $\varsigma_T(1)$ for $e(g_1,g_2)$. At the start of the build phase, the simulator randomly selects α,b from $\mathbb{Z}^*_p$.The public parameters $\left(e(g_1,g_2)^{\alpha}, g_1^b, g_2^{(\alpha+\beta)/b}, e(g_1,g_2)^{\alpha r}, g_1^{br}, r, \left\{\{g_1^{a_{ij}}\}_{1\le j\le n_i}\right\}_{1\le i\le n}\right)$ are sent to the adversary.

Query for private key $O_{KeyGen}(L)$: The premise is that $\mathcal{A}$ cannot ask for the private key of L that satisfies the access structure P^*.

$O_{KeyGen}(L)$:First, $\mathcal{B}$ substitutes $e(g_1,g_2)^{\alpha}$ with $\langle\alpha,\varsigma_T(\alpha)\rangle$ and adds new tuples $\langle\alpha x_n,\varsigma_T(\alpha x_n)\rangle$ of $e(g_1,g_2)^{\alpha x_u}$ using the rules defined above and using variables x_u from the list L_{G_T}. Then, $\mathcal{B}$ adds tuples $(x_u, \{\{g_2^{\beta+a_{i,t_i}\lambda_i}, g_2^{\lambda_i}\}_{1\le j\le n_i}\}_{1\le i\le n})$ to update the list relevant to the private key, and β,λ_i are the new variables.

Ciphertext challenge: $\mathcal{A}$ selects two messages m_0,m_1 in the actual choice safety game. Challenger $\mathcal{B}$ selects $\sigma \xleftarrow{R} \{0,1\}$ to encrypt m_σ using P^*.The challenge ciphertext as follows.

The simulation starts by selecting a random $\hat{r}$, setting λ_i for each of the relevant attributes, and λ_i for the random selection in $\mathbb{Z}_p$. The simulation randomly selects a θ. The constructed ciphertext is as follows: $\hat{c} = e(g_1,g_2)^{\theta}$, $\hat{c}_0 = B\hat{r}$, $\hat{c}_{i,1} = g_1^{\hat{r}_i}$, and $\hat{c}_{i,j} = A_{i,j}^{\hat{r}_i} = g_1^{a_{i,j}\hat{r}_i}$. The challenge ciphertext is sent to the adversary.

For up to q times after the inquiry, $\mathcal{A}$ terminates and returns a guess $\sigma'\in\{0,1\}$ of σ. Next, $\mathcal{B}$ chooses a random value of $\sigma \xleftarrow{R} \{0,1\}$, and obtains the real challenge ciphertext in list V_{G_1} by $\hat{c} = m_b e(g_1,g_2)^{\alpha\hat{r}}$ instead of $\hat{c} = e(g_1,g_2)^{\theta}$.Finally, $\mathcal{B}$ returns to the list of all tuples to update $\mathcal{A}$.

The challenger's task is to distinguish ciphertext $\hat{c}$ as $m_0 e(g_1,g_2)^{\alpha\hat{r}}$ or $m_1 e(g_1,g_2)^{\alpha\hat{r}}$.We now modify the game to distinguish $e(g,g)^{\alpha\hat{r}}$ from $e(g,g)^{\theta}$. Here, θ is randomly selected from $\mathbb{Z}_p$. If the game is not modified, and assuming that the opponent has an ε advantage, then, in the modified game, any adversary has at least an $\varepsilon/2$ advantage. It can be seen in two cases. One is that the adversary must distinguish $m_0 e(g_1,g_2)^{\alpha\hat{r}}$ from $e(g_1,g_2)^{\theta}$, and the other is to distinguish between $m_1 e(g_1,g_2)^{\alpha\hat{r}}$ and $e(g_1,g_2)^{\theta}$. It is obvious that the probabilities of the two are equal. We need to calculate the advantage that the adversary wins the game in the modified game.

Next, a detailed analysis of $\mathcal{B}$ is given. We note that $\mathcal{B}'$s simulation is perfect if there is no unexpected collision. The collision is for two different polynomials of $F_{\tau,l},F_{\tau,l'}(\tau\in\{1,2,T\})$ for some l,l', and all the random strings that encode the corresponding difference are not equal to 0. Therefore, $F_{\tau,l}-F_{\tau,l'} = 0$.This unexpected collision occurs in the following two situations:

Before the substitution. In this scenario, using theorem [37,38], the probability of an unexpected collision occurring in list V_{G_1}, V_{G_2}, V_{G_T} is at most $O(q^2/p)$.

After the substitution. It is impossible to have a new equation that can be created between polynomials $F = F_{k,l}-F_{k,l'}$, even if $\mathcal{B}$ is replaced by $\alpha\hat{r}$ for θ in the simulation. It is emphasized that adversary $\mathcal{A}$ cannot structure a query for a nonzero $F = F_{k,l}-F_{k,l'}$, and $F = 0$ after the substitution.

Note: If the adversary asks for the private key that satisfies the attributes of the access policy, the simulator does not give the appropriate private key. If the adversary already has the appropriate private key to access the structure, the game is terminated.

Table 1. Computational complexity.

Algorithm	Computational complexity			
	[10]	[13]	[18]	Our scheme
Establish	$3E_1$	$\left(\sum_{i=1}^{n} n_i + 1\right)E_1 + E_T$	$3E_1+E_T$	$\left(\sum_{i=1}^{n} n_i + 2\right)E_1 + E_2 + 2E_T$
KeyGen	$(2n+2)E_1$	$(2n+2)E_2+E_T$	$3nE_1$	$2nE_2+E_T$
Encrypted ciphertext	×	×	$(3n+1)E_1+(2n+1)E_T$	$(2n+1)E_1+E_T$
Encrypted index	$(2n+3)E_1$	$(2n+1)E_1+2E_T$	×	$2nE_1$
Token	$(2n+2)E_1$	$(2n+1)E_2$	×	$(2n+1)E_2$
Search	$(2n+2)P$	$(2n+1)P+E_T$	×	$(2n+1)P+E_T$
Decrypt	×	×	$2nP$	$(2n+1)P+E_T$

https://doi.org/10.1371/journal.pone.0206126.t001

θ is only included in the $e(g_1,g_2)^{\theta}$ in $\mathbb{G}_T$.We note that $F = F_{k,l}-F_{k,l'}$. If $F = 0$ after the substitution, then $F = \rho\alpha\hat{r} - \rho\theta$, where ρ is a constant. Note that $F \neq 0$, $F + \rho\theta = \rho\alpha\hat{r}$. We can increase this inquiry to the artificial adversary. We will prove that the adversary cannot construct a query of $e(g_1, g_2)^{\rho\alpha\hat{r}}$.

The only way the adversary can get $\alpha\hat{r}$ is through pair $\hat{c}_0, k_0$.Since $\hat{c}_0 = B\hat{r} = g_1^{b\hat{r}}$, $k_0 = g_2^{(\alpha+\beta)/b}$ obtains $\alpha\hat{r} + \beta\hat{r}$, the adversary needs to eliminate $\beta\hat{r}$. To get $\beta\hat{r}$,the adversary needs to combine $K_{i,1}, \hat{c}_{i,1}$ and $K_{i,2}, \hat{c}_{i,2}$. Note that $K_{i,1} = g_2^{\beta+a_{i,t_i}\lambda_i}, \hat{c}_{i,1} = g_1^{\hat{r}_i}$. Adversary $\mathcal{A}$ will get the $\beta\hat{r}_i + a_{i,t_i}\lambda_i\hat{r}_i$ query and wants to eliminate $\rho' a_{i,t_i}\hat{r}_i$.As $K_{i,2} = g_2^{\lambda_i}, \hat{c}_{i,2} = A_{i,j}^{\hat{r}_i} = g_1^{a_{i,t_i}\hat{r}_i}$, $\mathcal{A}$ will obtain $a_{i,t_i}\hat{r}_i\lambda_i$. We know that $\hat{r} = \sum_{r=1}^{n} \hat{r}_i$. However, we know that $\mathcal{A}$ cannot ask the private key of L that satisfies the access structure P^*. Therefore, there exists a $a_{i',t_i}\hat{r}_{i'}\lambda_{i'}$ that cannot be constructed in the above way since there are some attributes i' that belong to L_j and L does not satisfy policy P^*. Therefore, the adversary cannot get $\beta\hat{r}_{i'}$.We know that $\hat{r} = \sum_{r=1}^{n} \hat{r}_i$; therefore, the adversary cannot obtain the value $\beta\hat{r}$.

6. Performance evaluation

The analysis of computational complexity: As our scheme is based on bilinear model, the computational complexity of the proposed scheme mainly comes from the pairing operation and group exponentiation operations in each group, ignoring all multiplication and hashing operations. The pairing operation is denoted by P. The group exponentiation operations in each group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are represented by E_1,E_2,E_T. The implementation uses the Pairing Based Cryptography (PBC) library[39]. The computational complexity of the proposed scheme with some existing latest similar schemes is analyzed in Table 1. In the scenario, we suppose that there are n attributes. The i-th attribute has n_i possible values such that $i = 1,\cdots,n$.

Functional analysis: A functional comparison of our scheme with some existing schemes is illustrated in Table 2. It includes hidden access structures, multi-data owners, encrypted

Table 2. Functional comparison.

Scheme	Hidden policy	Multi owners	Encrypt		Update
Miao Y[10]	×	√	×	√	×
Yang K[12]	×	×	√	×	√
Qiu S [13]	√	×	×	√	×
Zhong H[18]	√	×	√	×	√
Our scheme	√	√	√	√	√

https://doi.org/10.1371/journal.pone.0206126.t002

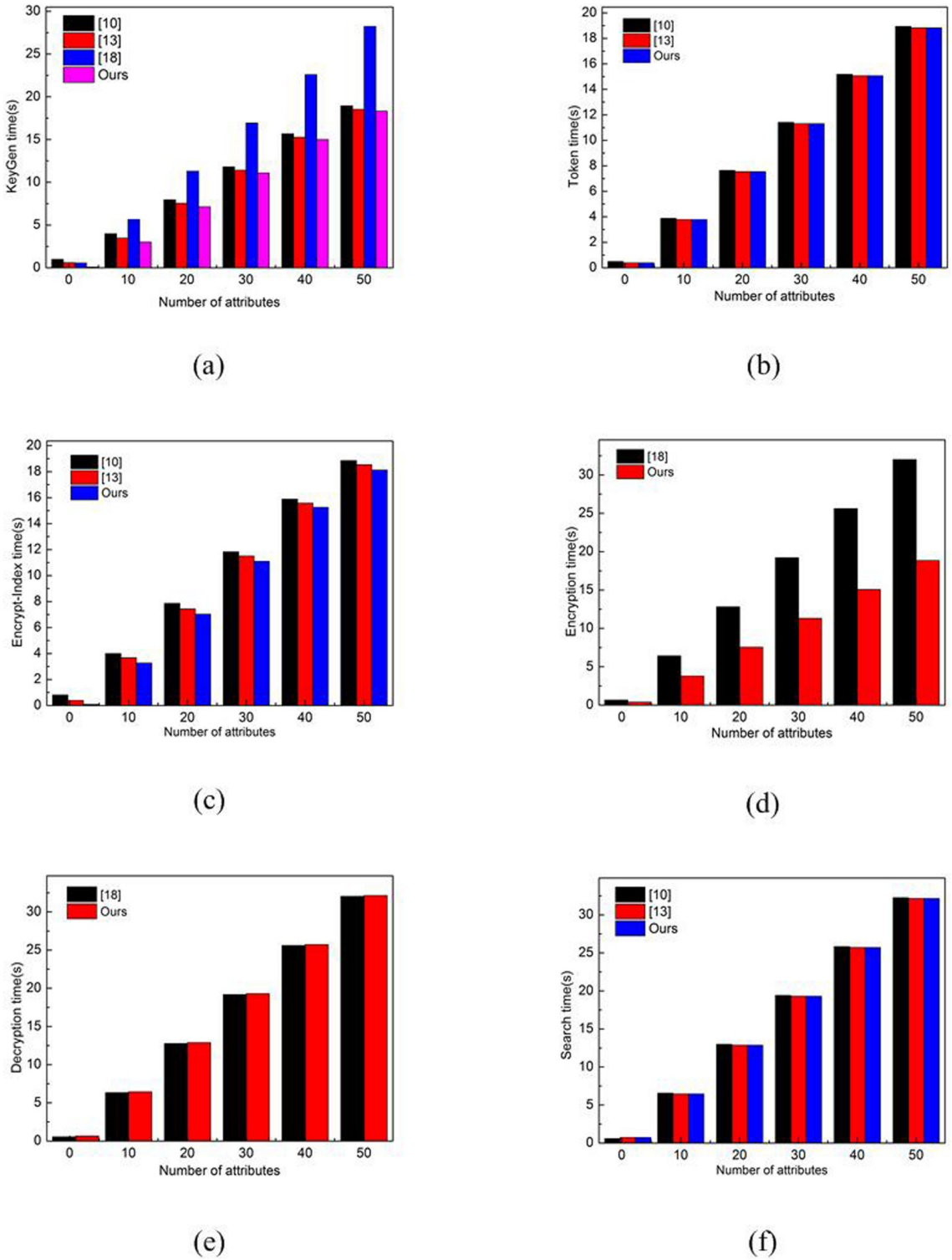

Fig 2. Computation comparison.

https://doi.org/10.1371/journal.pone.0206126.g002

messages, keyword search, and attribute changes, from which we can see that our scheme is fully functional.

The actual analysis: The actual execution time of each algorithm in the simulation experiments are as follows. We let n range from 1 to 100 in the access structure where n is the number of involved attributes. $n_i = 10$, where n_i is the possible values of the i-th attribute. We list a comparison of the average computation time for each algorithm in the scheme with the algorithm in [10, 13, 18] in Fig 2.

From the experimental results of computation comparison shown in Fig 2, we can see that as the increase of the number of attributes, the computation times of private key generating, encryption-index time generating and encryption ciphertext time are slightly better than these schemes [10,13,18]. In our scheme, the computation times of token time generating and keyword search are close to these schemes in [10,13,18].We notice that the decryption time of our scheme is a little more than that in [18], this is caused by the fact that our scheme is multi owners while the scheme in [18] is a single data owner. The scheme [18] cannot implement the search function, and the schemes [10,13] cannot achieve encryption ciphertext and decryption function. So our scheme is much practical than the schemes in [10,13,18].

7. Conclusions

In this paper, we present a new keyword searchable attribute-based encryption scheme with a hidden access strategy and attribute revocation. The encrypted ciphertext are outsourced to the cloud. The hidden strategy can better secure the users' privacy. Our proposed scheme is a fully functional scheme that addressed the keyword search problem and the attribute updating problem. Theoretical analysis, complexity calculation and practical operations show that our scheme is effective and practical. Of course, the scheme also has several short comings. The security of the scheme is demonstrated under the general bilinear group model, and it would be considerably better in the standard model.

Supporting information

S1 File. The runtime of cryptographic operations.
(DOCX)

Acknowledgments

Thanks to the anonymous reviewer for their useful comments.

Author Contributions

Software: Yaling Zhang.

Writing – original draft: Tingting Gao.

Writing – review & editing: Shangping Wang.

References

1. Jiang D, Xu Z, Liu J, et al. An optimization-based robust routing algorithm to energy-efficient networks for cloud computing[J]. Telecommunication Systems, 2016, 63(1):89–98.
2. Siddiqui Z, Abdullah A H, Khan M K, et al. Smart environment as a service: three factor cloud based user authentication for telecare medical information system [J]. Journal of Medical Systems, 2014, 38 (1):1–14. https://doi.org/10.1007/s10916-013-0001-1
3. Waqar A, Raza A, Abbas H, et al. A framework for preservation of cloud users' data privacy using dynamic reconstruction of metadata[J]. Journal of Network & Computer Applications, 2013, 36(1):235–248.

4. Lin H Y, Hsieh M Y, Li K C. Secured map reduce computing based on virtual machine using threshold secret sharing and group signature mechanisms in cloud computing environments[J]. Telecommunication Systems, 2015, 60(2):1–11.
5. Sahai A, Waters B. Fuzzy Identity-Based Encryption [M]. Advances in Cryptology–EUROCRYPT 2005. Springer Berlin Heidelberg, 2005:457–473.
6. Bethencourt J, Sahai A, Waters B. Ciphertext-Policy Attribute-Based Encryption [C]. IEEE Computer Society, 2007:321–334.
7. Ling C, Newport C. Provably Secure Ciphertext Policy ABE[C]. Proceedings of the 14th ACM Conference on Computer and Communications Security. ACM, 2007:456–465.
8. Su J S, Cao D, Wang X F, et al. Attribute-Based Encryption Schemes[J]. Journal of Software, 2011, 22 (6):1299–1315.
9. Goyal V, Pandey O, Sahai A, et al. Attribute-based encryption for fine-grained access control of encrypted data[C]. ACM Conference on Computer and Communications Security. ACM, 2006:89–98.
10. Yang K, Jia X. DAC-MACS: Effective Data Access Control for Multi-Authority Cloud Storage Systems [C]. IEEE INFOCOM. IEEE, 2014:2895–2903.
11. Song D X, Wagner D, Perrig A. Practical Techniques for Searches on Encrypted Data [C]. Proceeding 2000 IEEE Symposium on Security and Privacy. S&P. IEEE, 2000:44–55.
12. Miao Y, Ma J, Liu X, et al. m^2-ABKS: Attribute-Based Multi-Keyword Search over Encrypted Personal Health Records in Multi-Owner Setting [J]. Journal of Medical Systems, 2016, 40(11):246. https://doi.org/10.1007/s10916-016-0617-z PMID: 27696175
13. Qiu S, Liu J, Shi Y, et al. Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack[J]. Science China Information Sciences, 2017, 60(5):052105.
14. Zheng Q, Xu S, Ateniese G. VABKS: Verifiable attribute-based keyword search over outsourced encrypted data [C]. In: Proceedings of IEEE Conference on Computer Communications, INFOCOM, Toronto, 2014. 522–530.
15. Sun W, Yu S, Lou W, et al. Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud [C]. In: Proceedings of IEEE Conference on Computer Communications, INFOCOM, Toronto, 2014. 226–234.
16. Tang Y, Liu L. Privacy-Preserving Multi-Keyword Search in Information Networks [J]. IEEE Transactions on Knowledge & Data Engineering, 2015, 27(9):2424–2437.
17. Xia Z, Wang X, Sun X, et al. A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data[J]. IEEE Transactions on Parallel & Distributed Systems, 2016, 27(2):340–352.
18. Zhong H, Zhu W, Xu Y, et al. Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage[J]. Soft Computing, 2016(1):1–9.
19. Guo C, Zhuang R, Jie Y, et al. Fine-grained Database Field Search Using Attribute-Based Encryption for E-Healthcare Clouds[J]. Journal of Medical Systems, 2016, 40(11):235. https://doi.org/10.1007/s10916-016-0588-0 PMID: 27653042
20. Fan Y, Liu Z. Verifiable Attribute-Based Multi-keyword Search over Encrypted Cloud Data in Multi-owner Setting[C]. IEEE Second International Conference on Data Science in Cyberspace. IEEE, 2017:441–449.
21. Guo Z, Zhang H, Sun C, et al. Secure multi-keyword ranked search over encrypted cloud data for multiple data owners[J]. Journal of Systems & Software, 2018, 137:380–395.
22. Wang H, Zheng Z, Wu L, et al. New directly revocable attribute-based encryption scheme and its application in cloud storage environment[J]. Cluster Computing, 2017, 20(3):2385–2392.
23. Lai J, Deng R H, Li Y. Expressive CP-ABE with partially hidden access structures[C]. ACM Symposium on Information, Computer and Communications Security. ACM, 2012:18–19.
24. Nishide T, Yoneyama K, Ohta K. Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures[M]. Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2008:111–129.
25. Qian H, Li J, Zhang Y, et al. Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation[J]. International Journal of Information Security, 2015, 14(6):487–497.
26. Li X, Gu D, Ren Y, et al. Efficient Ciphertext-Policy Attribute Based Encryption with Hidden Policy[J]. 2012, 7646:146–159.
27. Tian Y, Peng Y, Peng X, et al. An Attribute-Based Encryption Scheme with Revocation for Fine-Grained Access Control in Wireless Body Area Networks [J]. International Journal of Distributed Sensor Networks, 2014, 25(4):820–835.

28. Xia Z, Zhang L, Liu D. Attribute-Based Access Control Scheme with Efficient Revocation in Cloud Computing [J]. China Communication, 2016, 13(7):92–99.
29. Chen J, Dai S, Song Z. Efficient decentralized attribute-based access control for cloud storage with user revocation[C]. IEEE International Conference on Communications. IEEE, 2014:3782–3787.
30. Li X, Tang S, Xu L, et al. Two-Factor Data Access Control With Efficient Revocation for Multi-Authority Cloud Storage Systems[J]. IEEE Access, 2017, PP (99):1–1.
31. Liu Z, Jiang Z L, Wang X, et al. Practical attribute-based encryption: Outsourcing decryption, attribute revocation and policy updating[J]. Journal of Network & Computer Applications, 2018, 108: 112–123.
32. Hur J, Noh D K. Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems [J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(7):1214–1221.
33. Pirretti M, Traynor P, Mcdaniel P, et al. Secure Attribute-Based Systems [J]. Journal of Computer Security, 2010, 18(5):799–837.
34. Chow S M. A Framework of Multi-Authority Attribute-Based Encryption with Outsourcing and Revocation[C]. ACM on Symposium on Access Control MODELS and Technologies. ACM, 2016:215–226.
35. Han Y, Di J, Yang X. The Revocable Attribute Based Encryption Scheme for Social Networks[C]. International Symposium on Security and Privacy in Social Networks and Big Data. IEEE, 2016:44–51.
36. Boneh D, Boyen X, Goh E J. Hierarchical identity based encryption with constant size ciphertext [M]. Advances in Cryptology–EUROCRYPT 2005. Springer Berlin Heidelberg, 2005:440–456.
37. Schwartz J T. Fast probabilistic algorithms for verification of polynomial identities [M]. Symbolic and Algebraic Computation. Springer Berlin Heidelberg, 1979:10–1145.
38. 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.
39. Duquesne S, Lange T. Pairing-based cryptography. Math.iisc.ernet.in, 2005, volume 22(3):573–590.