RESEARCH ARTICLE

Attribute-based encryption scheme with multi-keyword search and supporting attribute revocation in cloud storage

Shangping Wang¹, Lisha Yao^{1*}, Yaling Zhang²

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

* yaolishagh@163.com

OPEN ACCESS

Citation: Wang S, Yao L, Zhang Y (2018) Attribute-based encryption scheme with multi-keyword search and supporting attribute revocation in cloud storage. PLoS ONE 13(10): e0205675. <https://doi.org/10.1371/journal.pone.0205675>

Editor: Muhammad Khurram Khan, King Saud University, SAUDI ARABIA

Received: June 8, 2018

Accepted: September 29, 2018

Published: October 12, 2018

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

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

Funding: This research is supported by the National Natural Science Foundation of China (No. 61572019, 61173192, <http://www.nsf.gov.cn>) to S.W., the Key Project of Research Foundation of Natural Science Foundation of Shaanxi Province of China (NO.2016JZ001, <http://www.sninfo.gov.cn/>) to S.W. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Abstract

We propose an attribute-based encryption scheme with multi-keyword search and supporting attribute revocation in cloud storage environment, in which binary attributes and AND-gate access policy are used. Our proposal enjoys several advantages. Firstly, multi-keyword search is available, and only when a data user's attribute set satisfies access policy in keyword index, and keyword token generated by data user matches index successfully, then data user can obtain ciphertext containing keywords. In this way, more accurate keyword search is achievable. Secondly, the search privacy of data user is protected owing to cloud servers cannot obtain any knowledge of keywords which data user is interested in. Meanwhile, the ciphertext is able to be decrypted when data user's attribute set satisfies access policy specified in the ciphertext, which can both improve security of encryption and achieve secure fine-grained access control. Thirdly, the proposed scheme supports attribute revocation, in our scheme when a data user's attribute is revoked, the version number of attribute, non-revoked data users' secret keys and related ciphertexts will be updated, such that data user whose attribute is revoked does not decrypt updated ciphertext anymore. In addition, based on the assumption of decisional linear (DL) and decisional Diffie-Hellman (DDH), our scheme is proved to be secure against selectively chosen-keyword attacks and selectively chosen-plaintext attacks respectively, and it also ensures token privacy security.

Introduction

With the fast development of information technology, cloud storage now plays a very crucial role [1] in our daily life. For the sake of insuring data security, the important data that are uploaded to cloud server needs to be kept confidential, which requires data owners to encrypt private files before uploading. Meanwhile, it is also necessary to quickly find required files for data users by keyword searching from a vast amount of encrypted data. Therefore, in order to enable a secure keyword search and protect data user's search privacy, setting the keyword index of file is essential. That means that, although cloud server provides a search service, it does not know any information of keyword searching by data users. Consequently, it has

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

important theoretical value and practical significance to study secure and practical attribute-based encryption schemes that sustaining both attribute revocation and multi-keyword search.

In order to provide fine-grained access control for encrypted data, Sahai and Waters first proposed the notion of attribute-based encryption (ABE) in [2], which achieved one-to-many secure services based on public key encryption, and it ensured efficient encrypted access policy. Indeed, many attribute-based encryption (ABE) schemes have been presented, Goyal et al. [3] further defined the concept of attribute-based encryption (ABE). In general, attribute-based encryption (ABE) schemes are classified into two categories: One kind is key-policy attribute-based encryption (KP-ABE) [4–6], in which data user's secret key and ciphertext are relevant to access policy and attribute set, respectively. The other kind is ciphertext-policy attribute-based encryption (CP-ABE), which was first put forward by Bethencourt et al. in [7] that was proved to be safe under the general group model. In a CP-ABE scheme, the data user's secret key is related to attribute set and ciphertext is related to specific access policy. In 2008, Goyal et al. [8] proposed a CP-ABE scheme that was secure under the decisional bilinear Diffie-Hellman (DBDH) assumption. In 2012, Cheng et al. [9] presented a CP-ABE scheme in large universe set, which introduced attribute union, and it reduced storage and computational overhead of existing CP-ABE schemes. In 2016, Li et al. [10] added a testing phase to avoid unnecessary operation in their scheme and the proposal was proved to be safe under the decisional Diffie-Hellman (DDH) assumption. For resource-constrained devices, Odelu et al. [11] proposed a scheme that had constant size ciphertexts and secret keys. Recently, Shynu et al. [12] presented a notion of attributes hierarchical, constructed a separate database system for a meaningful data user's attribute set, and it solved the problem of attributes management.

Keyword searchable encryption is an effective solution to quickly find desired files from a vast amount of encrypted data managed in cloud servers. In 2003, Dan et al. [13] proposed a public key encryption with keyword search (PEKS) scheme. In 2010, Li et al. [14] presented a fuzzy keyword search for encrypted data scheme, which enhanced system usability by approximate matching of files and keywords. Subsequently, identity-based public key encryption and keyword searchable schemes were proposed in [15]. In 2010, Kamara et al. [16] put forward three models for data encryption and search. Since most of these schemes cannot support multi-keyword search, to address this problem, Cao et al. [17] proposed an encryption and multi-keyword sequence search scheme, which allowed multiple keywords in search phase, and it returned documents in relevant order. In 2012, the public key encryption and multi-keyword search scheme was proposed in [18]. In 2016, Miao et al. [19] presented an attribute-based multi-keyword search encryption scheme for personal health records in multi-owner environment, which provided an application direction for multi-keyword searchable encryption. In 2017, Huang et al. [20] proposed a multi-server multi-keyword searchable encryption scheme, which is proved to be secure against adaptive chosen keyword attack.

From a few years ago to today, many CP-ABE schemes that were sustaining attribute revocation have been mentioned. In 2010, Yu et al. [21] implemented a direct revocation of attributes in virtue of an agent, in which proxy key can be used to generate proxy re-encrypted ciphertext, and the scheme also had a capability of updating all corresponding secret keys of each legitimate data users. Afterwards, Yang et al. [22] presented a CP-ABE scheme that were supporting attribute revocation, in which attribute authority was responsible for updating ciphertexts and non-revoked data users' corresponding secret keys related to revoked attribute. In 2014, Xiong et al. [23] put forward a CP-ABE scheme that supporting universe attribute revocation, which was built on multiple minimum attribute sets of sharing re-encryption keys. When a universe attribute needs to be revoked, the cloud server performs the operation of re-encrypted ciphertext. In 2016, according to single authority attribute-based encryption (ABE) schemes, Chow [24] presented a scheme that was attribute-based encryption (ABE) with

supporting multi-authority and revocation. In 2017, Liu et al. [25] proposed an attribute-based encryption scheme that was sustaining both outsourced decryption algorithm and attribute revocation, which set a randomized version number for each attribute, thus attribute revocation is effectively implemented.

Recently, Wang et al. [26] proposed a CP-ABE scheme, which supported keyword searchable and attribute update in cloud storage. Our solution and scheme [26] are different in the following aspects: Firstly, access policy is different. The scheme [26] adopts linear secret sharing (LSSS) in the specific algorithm design, while our solution uses AND-gate access policy. Secondly, attribute update and revocation are different. Attribute update is used in [26], which updates a data user's original attribute to a new attribute and also updates the data user's secret key associated with the attribute. Our scheme is attribute revocation. Although scheme [26] also involves attribute revocation, it is still different from ours. We set the version number for each attribute, when the version number of revocation attribute changes, related ciphertexts and all non-revoked data users' secret keys are updated. Thirdly, the security proof method of schemes is different. The scheme [26] proves that the algorithms resist chosen-keyword attack based on the hard problem of bilinear Diffie-Hellman (BDH), and the scheme is proved to be secure against chosen-plaintext attack under the general bilinear group model. However, the security proof of our proposal is based on the hard problems. According to the assumption of decisional linear (DL) and decisional Diffie-Hellman (DDH), our scheme is proved to be secure against selectively chosen-keyword attack and selectively chosen-plaintext attack, respectively. At the same time, our solution is proved to enjoy token privacy security by using the unidirectional and collision-resistance of hash function. In addition, our scheme analyzes the forward and backward security for attribute revocation.

1.1 Our contributions

Considering that most of existing CP-ABE schemes cannot support attribute revocation and multi-keyword search, we present a CP-ABE scheme with multi-keyword search and supporting attribute revocation in cloud storage. The innovations can be summarized as follows:

1. Our scheme supports attribute revocation, when a data user's attribute is revoked, the version number of attribute, non-revoked data users' secret keys and related ciphertexts will be updated, such that data user whose attribute is revoked does not decrypt updated ciphertext anymore.
2. After data owners upload overall ciphertext of encrypted data to cloud server, the keyword search is used to quickly find required file. Compared to a single keyword search, the multi-keyword search is closer to real application. For the consideration of this issue, our scheme supports multi-keyword search.

Preliminaries

2.1 Bilinear map [27]

Let $\mathbb{G}, \mathbb{G}_T$ be two multiplicative cyclic groups with prime order p , g be a generator of group $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map with following properties:

1. Bilinearity: For any $a, b \in \mathbb{Z}_p$, exist $e(g^a, g^b) = e(g, g)^{ab}$.
2. Non-degeneracy: For $g \in \mathbb{G}$, such that $e(g, g) \neq 1$.
3. Computability: For any $u, v \in \mathbb{G}$, $e(u, v)$ is efficiently computed.

2.2 Access policy[28]

We denote by $U = \{attr_1, attr_2, \dots, attr_n\}$ the universe set of attributes, where n is the size of U , namely $|U| = n$. Let $Attr$ be attribute set of a data user. We introduce an n -bit string $\hat{v}_1 \hat{v}_2 \dots \hat{v}_n$ to express data user's attribute set $Attr = \hat{v}_1 \hat{v}_2 \dots \hat{v}_n$ as follows:

$$\begin{cases} \hat{v}_j = v_j : attr_j \in Attr \\ \hat{v}_j = \neg v_j : attr_j \notin Attr \end{cases} \quad j \in [1, n]$$

For example, let $n = 5$, suppose a data user's attribute set is $\{attr_1, attr_3, attr_5\}$, then it may be expressed as $Attr = v_1 \neg v_2 v_3 \neg v_4 v_5$.

We adopt AND-gate access policy and introduce an n -bit string $\tilde{v}_1 \tilde{v}_2 \dots \tilde{v}_n$ to express AND-gate access policy $S = \tilde{v}_1 \tilde{v}_2 \dots \tilde{v}_n$ as follows:

$$\begin{cases} \tilde{v}_j = v_j : attr_j \in S \\ \tilde{v}_j = \neg v_j : attr_j \notin S \end{cases} \quad j \in [1, n]$$

For example, let $n = 5$, suppose the access policy is $\{attr_1, attr_5\}$, then it may be expressed as $S = v_1 \neg v_2 \neg v_3 \neg v_4 v_5$.

For a data user's attribute set $Attr = \hat{v}_1 \hat{v}_2 \dots \hat{v}_n$ and an access policy $S = \tilde{v}_1 \tilde{v}_2 \dots \tilde{v}_n$. If for all $j \in [1, n]$, we have $\hat{v}_j \in S$, that is $\hat{v}_j = \tilde{v}_j$ ($\hat{v}_j$ represents the value of j -th attribute in data user's attribute set $Attr$, $\tilde{v}_j$ represents the value of j -th attribute in access policy S), we can say that the attribute set $Attr$ of data user satisfies access policy S . For convenience, we define a function $\gamma(Attr, S) \in \{0, 1\}$, when $\gamma(Attr, S) = 0$, it indicates that data user's attribute set $Attr$ does not satisfy access policy S . When $\gamma(Attr, S) = 1$, it indicates that the attribute set $Attr$ of data user satisfies access structure S .

2.3 Complexity assumption

In our proposal, the security depends on Decisional linear (DL) assumption [29] and Decisional Diffie-Hellman (DDH) assumption [30]. The specific description is:

Definition 1 (Decisional linear assumption). If for all polynomial-time adversary $\mathcal{A}$ who could successfully distinguish tuple $(g, f, h, f^{r_1}, g^{r_2}, h^{r_1+r_2})$ from tuple $(g, f, h, f^{r_1}, g^{r_2}, R)$ with a negligible advantage, and the advantage $ADV(\mathcal{A})$ of polynomial-time adversary $\mathcal{A}$ can be marked as

$$ADV(\mathcal{A}) = |\Pr[\mathcal{A}(g, f, h, f^{r_1}, g^{r_2}, h^{r_1+r_2}) = 1] - \Pr[\mathcal{A}(g, f, h, f^{r_1}, g^{r_2}, R) = 1]|$$

where $g, f, h, R \in \mathbb{G}$, $r_1, r_2 \in \mathbb{Z}_p^*$.

Definition 2 (Decisional Diffie-Hellman assumption). If for all polynomial-time adversary $\mathcal{A}$ who could successfully distinguish tuple $(g, g^{z_1}, g^{z_2}, g^{z_1 z_2})$ from tuple (g, g^{z_1}, g^{z_2}, Q) with a negligible advantage, and the advantage $ADV(\mathcal{A})$ of polynomial-time adversary $\mathcal{A}$ can be marked as

$$ADV(\mathcal{A}) = |\Pr[\mathcal{A}(g, g^{z_1}, g^{z_2}, g^{z_1 z_2}) = 1] - \Pr[\mathcal{A}(g, g^{z_1}, g^{z_2}, Q) = 1]|$$

where $g, Q \in \mathbb{G}$, $z_1, z_2 \in \mathbb{Z}_p^*$.

Our scheme

3.1 System model

The section contains overall framework of our scheme and the construction of solution.

Fig 1. A framework of scheme.

<https://doi.org/10.1371/journal.pone.0205675.g001>

3.1.1 System framework. Our scheme structure is shown in Fig 1. It contains following five entities:

Attribute Authority (AA): The AA is attribute authority, which is responsible for system's initial establishment and the local secret key generation of data user. Simultaneously, it distributes corresponding secret key according to attribute set for data user. When an attribute is revoked, AA generates an update key and completes partial secret key update.

Cloud Server (CS): The CS stores ciphertext which containing encrypted files and keyword indexes generated by data owners. Afterwards, when a data user tends to search ciphertext, CS

completes a matching of data user's token and keyword index. If matching succeeds, it sends ciphertext to data user. Additionally, in attribute revocation phase, CS is responsible for updating ciphertext.

Key Generation Server (KGS): The KGS generates data user's partial secret key, namely outsourced secret key, which effectively reduces the computational burden of AA. Besides, KGS is responsible for completing the update of outsourced secret key when attribute revocation happens.

Data Owner (DO): The DO encrypts keyword set and file to be shared, uploads ciphertext to cloud server. Only attribute set of data user who wants to access data satisfies access structure in ciphertext, that is $\gamma(Attr, S) = 1$, the encrypted data will be shared with data user. To be specific, the encryption operation to be completed by DO includes: the keyword index generation, the file encryption, and the encryption of key for encrypted file, hence ciphertext consists of three parts.

Data User (DU): When data user's attribute set satisfies access structure in ciphertext, then data user DU is able to access encrypted data and recover original plaintext. Specifically, DU generates desired keyword token and sends to cloud server CS, the CS makes a matching between search token and keyword index, if matching succeeds, DU can download corresponding ciphertext. In other words, DU is responsible for generating keyword token which he is interested in and decrypting ciphertext.

In particular, PK is public parameter published by attribute authority. Cph is ciphertext encrypted by data owner, and it includes three parts: 1) The keyword set WD is encrypted to generate keyword index, namely $Index$; 2) The encryption key ck is encrypted to obtain ciphertext CT' ; 3) The file M is symmetrically encrypted by using encryption key ck to gain $E_{ck}(M)$. At last, data owner uploads ciphertext Cph to cloud server. Hereafter, OK is used to denote intermediate key, which is generated by attribute authority according to attribute set of data user, and it is sent to key generation server. The key generation server calculates data user's partial secret key based on OK , namely outsourced secret key SK_1 , which is sent to attribute authority. Followed by, the attribute authority generates data user's local secret key SK_2 , and obtains data user's secret key $SK = (SK_1, SK_2)$, which is forwarded to data user. Tok is used to denote token generated by data user based on desired keyword set, which is used to match with keyword index. The cloud server executes search algorithm, if token matches index successfully, the cloud server transmits stored ciphertext Cph to data user. Then data user first decrypts ciphertext Cph with secret key SK to gain encryption key ck , and then symmetrically decrypts ciphertext with ck to obtain file M . In addition, when an attribute needs to be revoked, attribute authority sends instructions to cloud server to update ciphertext.

3.1.2 Standard definitions of our scheme. Let U be a universe set of attributes, S be an access policy, $Attr$ be a data user's attribute set. Attribute-based encryption scheme with multi-keyword search and supporting attribute revocation in cloud storage consists of 9 algorithms:

Setup(U, l) $\rightarrow PK, MSK$: The setup algorithm is executed by attribute authority AA. It inputs universe set of attributes U and security parameter l , and outputs system public key PK and master secret key MSK .

Encrypt: The encryption algorithm is run by data owner DO, including the following two parts:

- i) **Keyword-Encrypt**(PK, S, WD) $\rightarrow Index$: This algorithm takes as inputs access policy S , public key PK and keyword set WD , and it then outputs the ciphertext $Index$ of keyword set WD .
- ii) **Encryption key-Encrypt**(PK, S, ck) $\rightarrow CT'$: DO first symmetrically encrypts file M rely on encryption key ck to obtain $E_{ck}(M)$, and then encrypts encryption key ck as follows: This

algorithm makes access policy S , public key PK and encryption key ck as input, it generates ciphertext CT' .

Finally, DO uploads overall ciphertext $Cph = (Index, E_{ck}(M), CT')$ to cloud server CS.

In-KeyGen($PK, MSK, Attr$) $\rightarrow OK$: The intermediate key generation algorithm is executed by attribute authority AA. It makes public key PK , master secret key MSK and data user's attribute set $Attr$ as input, then produces intermediate key OK , and sends it to key generation server KGS.

Out-KeyGen(PK, OK) $\rightarrow SK_1$: The outsourced secret key generation algorithm is executed by key generation server KGS. It takes public key PK and intermediate key OK as input, and outputs outsourced secret key SK_1 , then returns SK_1 to attribute authority AA.

KeyGen($PK, MSK, Attr, SK_1$) $\rightarrow SK$: The secret key generation algorithm is executed by attribute authority AA. It makes public key PK , master secret key MSK , data user's attribute set $Attr$ and outsourced secret key SK_1 as input, then generates data user's secret key SK , and transmits it to data user DO.

TokenGen(PK, SK_1, WD') $\rightarrow Tok$: The token generation algorithm is executed by data user DU. It makes public key PK , outsourced secret key SK_1 and desired keyword set WD' as input, and it outputs token Tok , then forwards to cloud server CS.

Search($Index, Tok$) $\rightarrow \{0, 1\}$: The search algorithm is executed by cloud server CS. It makes index and token as input, outputs 1 if index and token can match successfully, then cloud server CS sends ciphertext Cph to data user DU, otherwise outputs 0 and terminates.

Decrypt(SK_2, CT') $\rightarrow ck$: The decryption algorithm is executed by data user DU. It makes local secret key SK_2 and ciphertext CT' of encryption key as input, generates encryption key ck , then it decrypts $E_{ck}(M)$ with ck , finally obtains file M .

Attribute revocation: The attribute revocation includes the following three aspects, note that only components related to revoked attribute will be updated.

- (i) The attribute authority AA takes charge of generating update key: AA generates a new version number of revoked attribute according to its old version number, then obtains update key. This algorithm makes update key, public key PK and master secret key MSK as input, outputs updated public key and updated master secret key.
- (ii) The non-revoked data user's secret key update: The attribute authority AA updates intermediate key OK and local secret key SK_2 with update key, sends updated intermediate key to key generation server KGS. Then, KGS completes the update of outsourced secret key SK_1 , and transmits to AA. In the end, AA returns updated secret key to non-revoked data user.
- (iii) The cloud server CS is in charge of updating ciphertext: The cloud server CS executes this algorithm. It makes update key and overall ciphertext as input, outputs updated keyword index and updated ciphertext of encryption key.

3.2 Security model

Under the cloud storage environment, we suppose that attribute authority and key generation server are all trusted. But cloud server is semi-trusted, such as it can execute protocols honestly but also attempt to gain extra information from the protocol.

3.2.1 Selectively secure game for chosen-keyword attacks. **Setup**: First of all, adversary $\mathcal{A}$ sends a challenge access policy S^* to challenger $\mathcal{C}$, the challenger $\mathcal{C}$ runs **Setup** algorithm to generate public key PK and master secret key MSK , while keeping MSK secret.

Phase 1: Before Phase 1 begins, the challenger $\mathcal{C}$ initializes an empty keyword set query list L_W , then adversary $\mathcal{A}$ issues the following polynomial times adaptive queries:

Outsourced secret key query: According to **In-KeyGen** algorithm and **Out-KeyGen** algorithm, the adversary $\mathcal{A}$ submits a query attribute set $Attr^*$ to challenger $\mathcal{C}$. If attribute set $Attr^*$ does not satisfy access policy S^* , the adversary $\mathcal{A}$ obtains outsourced secret key SK_1 , otherwise terminates.

Token query: The adversary $\mathcal{A}$ commits a query keyword set WD' . According to **TokenGen** algorithm, the challenger $\mathcal{C}$ inputs public key PK , outsourced secret key SK_1 and query keyword set WD' to gain token Tok . If query attribute set $Attr^*$ does not satisfy access policy S^* , then challenger $\mathcal{C}$ adds WD' to list L_W and sends Tok to adversary $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ randomly selects two keyword sets WD'_0 and WD'_1 that are not in list L_W . The challenger $\mathcal{C}$ throws a fair coin to choose $\xi \in \{0,1\}$, runs **Keyword-Encrypt** algorithm to gain the index of keyword set WD'_ξ and transmits it to adversary $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ repeats queries in Phase 1, but keyword set WD'_0 and WD'_1 can no longer be queried.

Guess. Finally, the adversary $\mathcal{A}$ gives a guess ξ' of ξ . If $\xi' = \xi$, $\mathcal{A}$ wins game.

The advantage of adversary $\mathcal{A}$ can be defined as

$$ADV(\mathcal{A}) = |\Pr[\xi' = \xi] - \frac{1}{2}|$$

Definition 3: If for all polynomial-time adversary who winning game with a negligible advantage, our scheme is called selectively secure against chosen-keyword attacks.

3.2.2 Token privacy game. To ensure the privacy of keyword, an adversary should not infer keyword information from token. In other words, if there is no polynomial-time adversary who can obtain keyword from token, the token privacy security can be guaranteed. The game is set up as follows:

Setup: The challenger $\mathcal{C}$ runs **Setup** algorithm, generates public key PK and master secret key MSK , while keeping MSK secret.

Phase 1: Similar to selectively secure game of chosen-keyword attacks, the challenger $\mathcal{C}$ initializes an empty key query list L_K , the adversary issues the following polynomial q_t times adaptive queries:

Outsourced secret key query: The adversary $\mathcal{A}$ selects a query attribute set $Attr$ for challenger $\mathcal{C}$. The challenger $\mathcal{C}$ outputs outsourced secret key SK_1 by running **In-KeyGen** algorithm and **Out-KeyGen** algorithm, then returns SK_1 to adversary $\mathcal{A}$, and the attribute set $Attr$ is added to list L_K .

Token query: The challenger $\mathcal{C}$ runs **TokenGen** algorithm based on public key PK , outsourced secret key SK_1 , and a query keyword set WD' given by adversary, then the challenger $\mathcal{C}$ sends token to adversary $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ submits an access policy S^* with the restriction that the attribute set $Attr$ in list L_K does not satisfy access policy S^* . Afterwards, the challenger $\mathcal{C}$ randomly chooses keyword set WD^* , encrypts it with S^* to obtain the index, and then selects an attribute set $Attr^*$ such that $Attr^*$ satisfies S^* . The challenger $\mathcal{C}$ executes **In-KeyGen** algorithm, **Out-KeyGen** algorithm and **TokenGen** algorithm to gain the token of keyword set WD^* , and transmits token to adversary $\mathcal{A}$.

Phase 2: Similar to Phase 1, but keyword set WD^* can no longer be inquired.

Guess: The adversary $\mathcal{A}$ gives a keyword set WD'' and forwards it to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ runs **Keyword-Encrypt** algorithm to get the index of keyword set WD'' , and makes a matching between the token of WD^* and the index of WD'' . If the result returned by **Search** algorithm is 1, then the adversary $\mathcal{A}$ wins game.

The advantage of adversary $\mathcal{A}$ can be defined at most

$$ADV(\mathcal{A}) = \frac{1}{|\Pi| - q_t} + \sigma$$

Because the adversary $\mathcal{A}$ wants to gain keyword information from the index, it is necessary to analyze the structure of index. In other words, the adversary $\mathcal{A}$ gets information from $H(w'_t)$ at most. Due to the unidirectional and collision-resistant properties of the hash function, we assume that the adversary $\mathcal{A}$ obtains the advantage of w' from $H(w'_t)$ is σ , where σ is a negligible probability under the security parameter l . Π is a keyword space for the selection of keyword set, $|\Pi|$ represents the size of keyword space and is large enough in practical applications. q_t denotes the number of inquiries about outsourcing private key and token in Phase 1, and q_t is finite size.

The advantage $ADV(\mathcal{A}) = \frac{1}{|\Pi| - q_t} + \sigma$ consists of two parts. $|\Pi| - q_t$ is the size of the remaining keyword set in the keyword space after the phase 1 inquiry. The probability that the adversary $\mathcal{A}$ guesses the encrypted keyword from the remaining keyword set is $\frac{1}{|\Pi| - q_t}$. σ is the probability that the adversary gets the keyword information from $H(w'_t)$. The advantage of adversary $\mathcal{A}$ is $\frac{1}{|\Pi| - q_t} + \sigma$ by summing two probabilities. Therefore, the advantage of adversary $\mathcal{A}$ can be defined at most $ADV(\mathcal{A}) = \frac{1}{|\Pi| - q_t} + \sigma$ which is a negligible advantage.

Definition 4: Our proposal is token privacy secure if all polynomial-time adversary have at most a negligible advantage in above game.

3.2.3 Selectively secure game for chosen-plaintext attacks. The section contains two indistinguishable games. Since the process of two games is similar, we only show the security proof of one of the games, the other game is described in detail in the specific security proof phase. The game is as follows:

Initialization: The adversary $\mathcal{A}$ commits two challenge access policies S_0^* and S_1^* .

Setup: The simulator $\mathcal{B}$ runs **Setup** algorithm, produces public key PK and master secret key MSK , while keeping MSK secret and sending PK to adversary $\mathcal{A}$.

Phase 1: The adversary $\mathcal{A}$ chooses a query attribute set $Attr$ to simulator $\mathcal{B}$.

Secret key query: If attribute set $Attr$ does not satisfy access policies S_0^* and S_1^* , that is $\gamma(Attr, S_0^*) = 0 \wedge \gamma(Attr, S_1^*) = 0$, then simulator $\mathcal{B}$ runs **KeyGen** algorithm to obtain local secret key SK_2 , and sends to adversary $\mathcal{A}$.

Challenge: For access policies S_0^* and S_1^* , the adversary $\mathcal{A}$ submits two equal-length encryption keys ck_1 and ck_2 that are used to encrypt file. Hereafter, the simulator $\mathcal{B}$ randomly throws a fair coin to select $b \in \{0, 1\}$ and runs **Encryption key-Encrypt** algorithm to gain the ciphertext CT' of encryption key ck_b , then transmits CT' to adversary $\mathcal{A}$.

Phase 2: Similar to Phase 1, the adversary $\mathcal{A}$ continues to query. Nevertheless, the restriction is $\gamma(Attr, S_0^*) = 0 \wedge \gamma(Attr, S_1^*) = 0$.

Guess. Finally, the adversary $\mathcal{A}$ makes a guess b' of b . If $b' = b$, $\mathcal{A}$ wins game.

The advantage of adversary $\mathcal{A}$ to win the game can be defined as

$$ADV(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

Definition 5: Our scheme is selectively secure against chosen-plaintext attacks if for all polynomial-time adversary who could win the game with a negligible advantage.

Concrete construction

In this section, we take into consideration that most of CP-ABE schemes only support a few functions, which has its limitations for practical application. This motivates us to construct a scheme that supports attribute revocation and multi-keyword search. Meanwhile, our scheme is almost the same as some schemes in terms of calculation amount and calculation time, and even smaller and faster. The following is the specific structure of scheme:

Setup(U, l) $\rightarrow PK, MSK$: This algorithm takes universe set of attributes $U = \{attr_1, attr_2, \dots, attr_n\}$ and security parameter l as input, it selects a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}, \mathbb{G}_T$ are two multiplicative cyclic groups with prime order p , g is a generator of group $\mathbb{G}$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ is a one-way hash function. The algorithm chooses $a, b, c, \alpha, \{r_1, r_2, \dots, r_{2n}\}$ from $\mathbb{Z}_p$ and, $\{x_1, x_2, \dots, x_{2n}\}$ from $\mathbb{G}$ at random, then let $Y = e(g, g)^\alpha$. For each attribute $attr_j \in U (j \in [1, n])$, it picks $vk_j \in \mathbb{Z}_p$ as initial version number, then sets $PK_{j,1} = g^{vk_j}$. Since the attribute $attr_j$ has two different values v_j and $\neg v_j$, let $\{r_1, \dots, r_n\}$ and $\{x_1, \dots, x_n\}$ denote corresponding parameters when $attr_j$ is equal to v_j , $\{r_{n+1}, \dots, r_{2n}\}$ and $\{x_{n+1}, \dots, x_{2n}\}$ denote corresponding parameters when $attr_j$ is equal to $\neg v_j$. Thus, for all $j = 1, \dots, n$, when $attr_j$ is equal to v_j , we have $PK_{j,2} = PK'_{j,2} = u_j = g^{-vk_j r_j}$, when $attr_j$ is equal to $\neg v_j$, we have $PK_{j,2} = PK''_{j,2} = u_{j+n} = g^{-vk_j r_{j+n}}$. Similarly, the algorithm computes $PK_{j,3} = PK'_{j,3} = y_j = e(x_j, g)^{vk_j}$ if $attr_j$ is equal to v_j , it computes $PK_{j,3} = PK''_{j,3} = y_{j+n} = e(x_{j+n}, g)^{vk_j}$ if $attr_j$ is equal to $\neg v_j$. Afterwards, the algorithm randomly selects $\beta \in \mathbb{Z}_p$, sets $PK_{j,4} = X_j = g^{\frac{\beta}{vk_j}}$. The public attribute key PK_j is:

$$PK_{j,1} = g^{vk_j}$$

$$PK'_{j,2} = u_j = g^{-vk_j r_j}$$

$$PK''_{j,2} = u_{j+n} = g^{-vk_j r_{j+n}}$$

$$PK'_{j,3} = y_j = e(x_j, g)^{vk_j}$$

$$PK''_{j,3} = y_{j+n} = e(x_{j+n}, g)^{vk_j}$$

$$PK_{j,4} = X_j = g^{\frac{\beta}{vk_j}}$$

Where $j \in [1, n]$, $vk_j \in \mathbb{Z}_p$ denotes initial version number of attribute $attr_j$. For simplicity, let

$$PK_j = \{PK_{j,1}, PK_{j,4}, (PK'_{j,2}, PK'_{j,3}) = (u_j, y_j), (PK''_{j,2}, PK''_{j,3}) = (u_{j+n}, y_{j+n})\}_{j \in [1, n]}$$

Subsequently, the algorithm generates public key PK and master secret key MSK :

$$PK = (e, \mathbb{G}, \mathbb{G}_T, g, g^a, g^b, g^c, Y, H, \{PK_j | attr_j \in U\})$$

$$MSK = (a, b, c, \alpha, \beta, \{(r_i, x_i)\}_{i \in [1, 2n]}, \{MSK_j = vk_j | attr_j \in U\})$$

While keeping MSK secret.

Encrypt: This algorithm defines an access policy is $S = \tilde{v}_1 \tilde{v}_2 \cdots \tilde{v}_n$, denotes as

$$u'_j = \begin{cases} u_j, & \tilde{v}_j = v_j \\ u_{j+n}, & \tilde{v}_j = \neg v_j \end{cases} \quad j \in [1, n]$$

The encryption algorithm includes two steps: Step one is to encrypt keyword set, that is to generate an index of keyword set, step two is to encrypt encryption key.

(i) **Keyword-Encrypt**(PK, S, WD) $\rightarrow$ *Index*

This algorithm makes public key PK , access policy S and a keyword set $WD = \{w_1, w_2, \dots, w_r\}$ extracted from file M as input, where r is the size of WD , namely $|WD| = r$. It randomly

picks $t_1, t_2 \in \mathbb{Z}_p$, computes $\tilde{u} = g^{t_2} \prod_{j=1}^n u'_j$, then sets $W' = g^{ct_1}$ and $W_t = g^{a(t_1+t_2)} g^{bH(w_t)t_1}$,

where $w_t \in WD$, $t \in [1, r]$. Hereafter, the algorithm outputs the index of keyword set as

$$Index = (\tilde{u}, W', \{W_t\}_{t \in [1, r]})$$

(ii) **Encryption key-Encrypt**(PK, S, ck) $\rightarrow$ CT'

Before uploading a file M , the algorithm encrypts the file as follows:

- ① It selects an encryption key ck from key space, and symmetrically encrypts the file M with encryption key ck to obtain $E_{ck}(M)$.
- ② It sets an access structure $S = \tilde{v}_1 \tilde{v}_2 \cdots \tilde{v}_n$, encrypts ck and outputs the ciphertext of encryption key ck through the following steps.

This algorithm makes public key PK , access policy S and encryption key ck as input. It chooses $s \in \mathbb{Z}_p$ at random, computes $C = ck \cdot Y^s$ and $C_1 = g^s$, then picks up random value $s_j \in \mathbb{Z}_p$

such that $s = \sum_{j=1}^n s_j$, it computes $C_{j,1} = X_j^{s_j}$ and $C_{j,2} = u'^{s_j}$. Consequently, the ciphertext of encryption key ck as follows:

$$CT' = (C, C_1, \{(C_{j,1}, C_{j,2})\}_{j \in [1, n]})$$

Finally, the algorithm outputs the overall ciphertext $Cph = (Index, E_{ck}(M), CT')$.

In-KeyGen($PK, MSK, Attr$) $\rightarrow$ *OK*: This algorithm makes public key PK , master secret key MSK and an attribute set $Attr$ as input. It sets $v = g^{ac}$, computes $\sigma_j (j \in [1, n])$ according to attribute set $Attr = \hat{v}_1 \hat{v}_2 \cdots \hat{v}_n$. Let

$$\sigma_j = \begin{cases} (x_j v^{r_j})^{v^{k_j}}, & \hat{v}_j = v_j \\ (x_{j+n} v^{r_{j+n}})^{v^{k_j}}, & \hat{v}_j = \neg v_j \end{cases} \quad j \in [1, n]$$

At last, this algorithm outputs the intermediate key as follows:

$$OK = (Attr, v, \{\sigma_j\}_{j \in [1, n]})$$

Out-KeyGen(PK, OK) $\rightarrow$ SK_1 : This algorithm makes public key PK and intermediate key OK as input. It first computes $\tilde{\sigma} = \prod_{j=1}^n \sigma_j$, then computes $\tilde{y} = \prod_{j=1}^n y'_j$ based on attribute set

$Attr = \hat{v}_1 \hat{v}_2 \cdots \hat{v}_n$, where

$$y'_j = \begin{cases} y_j, & \hat{v}_j = v_j \\ y_{j+n}, & \hat{v}_j = \neg v_j \end{cases} \quad j \in [1, n]$$

Afterwards, the algorithm outputs the outsourced secret key as

$$SK_1 = (v, \tilde{\sigma}, \tilde{y})$$

KeyGen($PK, MSK, Attr, SK_1$) $\rightarrow SK$: This algorithm makes public key PK , master secret key MSK , attribute set $Attr$ and outsourced secret key SK_1 as input. It randomly picks $u, \lambda_j \in \mathbb{Z}_p$, where $j \in [1, n]$, the algorithm lets $D_1 = g^{\alpha + \beta u}$ and $D_{j,1} = X_j^{\lambda_j}$, it computes $D_{j,2} = g^{vk_j(u - r_{j+n}\lambda_j)}$ when $\hat{v}_j = v_j$, otherwise computes $D_{j,2} = g^{vk_j(u - r_{j+n}\lambda_j)}$, then the local secret key as

$$SK_2 = (D_1, \{(D_{j,1}, D_{j,2})\}_{j \in [1, n]})$$

Finally, the algorithm outputs data user's secret key $SK = (SK_1, SK_2)$.

TokenGen(PK, SK_1, WD') $\rightarrow Tok$: This algorithm makes public key PK , outsourced secret key SK_1 and a keyword set $WD' = \{w'_1, w'_2, \dots, w'_d\}$ of interest as input, where the size of WD' is d , namely $|WD'| = d$. It chooses random value $z \in \mathbb{Z}_p$, computes $Tok_1 = \prod_{t=1}^d (g^a g^{bH(w'_t)})^z$ and $Tok_2 = g^{cz}$. Therefore, the algorithm generates a token of keyword set for data user as follows:

$$Tok = (v^z, \tilde{\sigma}^z, \tilde{y}^z, Tok_1, Tok_2)$$

Search($Index, Tok$) $\rightarrow \{0, 1\}$: Taking as inputs the index relevant to access policy S and the token relevant to attribute set $Attr$, this algorithm is executed by cloud sever to test whether there is a matching between the index and the token. In other words, the cloud server determines whether the following equation holds:

$$\frac{e\left(\prod_{l=1}^d W_{t_l}, Tok_2\right)}{e(W', Tok_1)} \stackrel{?}{=} \frac{e(\tilde{u}, v^z) e(\tilde{\sigma}^z, g)}{\tilde{y}^z} \quad (1)$$

In above equation, it involves a matching of the index and the token. In the index generation phase, the data owner encrypts r keywords to obtain $\{W_t\}_{t \in [1, r]}$. In the token generation phase, the data user generates a token for d keywords which he is interested in, and computes Tok_1 , particularly $d \leq r$. In order to make the Eq (1) can be calculated, the cloud server is to arbitrarily select d components from $\{W_t\}_{t \in [1, r]}$ and execute multiplication operations. In accordance with the theory of probability and mathematical statistics, the total number of such choices is $C_r^d = \frac{r(r-1)(r-2) \cdots (r-d+1)}{d!}$. Hereafter, the cloud server matches the multiplication of $W_{t_l} (l \in [1, d])$ with Tok_1 . In the matching of C_r^d times, as long as there is one successful match, it demonstrates that the above equation holds and the search succeeds.

If and only if the Eq (1) holds, the cloud server returns 1 and transmits overall ciphertext $Cph = (Index, E_{ck}(M), CT')$ to data user, otherwise returns 0.

Decrypt(SK_2, CT') $\rightarrow ck$: This algorithm makes data user's local secret key SK_2 and ciphertext CT' of encryption key as input. If data user's attribute set $Attr$ satisfies access policy S

embedded in the ciphertext, the algorithm decrypts CT' to obtain encryption key ck as follows:

$$ck = \frac{C \prod_{j=1}^n e(C_{j,1}, D_{j,2})}{e(C_1, D_1) \prod_{j=1}^n e(C_{j,2}, D_{j,1})} \quad (2)$$

Finally, a symmetric decryption algorithm is used to decrypt $E_{ck}(M)$ with the encryption key ck to gain the file M .

Correctness: Only when the two conditions $\gamma(Attr, S) = 1$ and $WD' \subseteq WD$ are both satisfied, the search can succeed. The following is the verification process of Eq (1):

$$\begin{aligned} \frac{e\left(\prod_{t=1}^d W_{t_i}, Tok_2\right)}{e(W', Tok_1)} &= \frac{e\left(\prod_{t=1}^d g^{a(t_1+t_2)} g^{bH(w_t)t_1}, g^{cz}\right)}{e\left(g^{ct_1}, \prod_{t=1}^d g^{az} g^{bzH(w'_t)}\right)} \\ &= \frac{e(g, g)^{at_1cz} e(g, g)^{at_2cz} e(g, g)^{bt_1cz \sum_{t=1}^d H(w_t)}}{e(g, g)^{ct_1bz \sum_{t=1}^d H(w'_t)}} \\ &= e(g, g)^{t_2acz} \\ \frac{e(\tilde{u}, v^z) e(\tilde{\sigma}^z, g)}{\tilde{y}^z} &= \frac{e\left(g^{t_2} \prod_{j=1}^n u'_j, g^{acz}\right) e\left(\prod_{j=1}^n \sigma_j^z, g\right)}{\prod_{j=1}^n y_j^z} \\ &= \frac{e\left(g^{t_2} \prod_{j=1}^n g^{-vk_j r_i}, g^{acz}\right) e\left(\prod_{j=1}^n x_i^{vk_j z} g^{vk_j acr_i z}, g\right)}{\prod_{j=1}^n e(x_i, g)^{vk_j z}} \\ &= \frac{e(g, g)^{t_2acz} \prod_{j=1}^n e(g, g)^{-vk_j acr_i} \prod_{j=1}^n e(x_i, g)^{vk_j z} \prod_{j=1}^n e(g, g)^{vk_j acr_i z}}{\prod_{j=1}^n e(x_i, g)^{vk_j z}} \\ &= e(g, g)^{t_2acz} \end{aligned}$$

Where $i \in [1, 2n]$.

Only when the attribute set of data user satisfies access structure, that is $\gamma(Attr, S) = 1$, encryption key ck is able to be computed, and the correctness of Eq (2) is verified as follows:

$$\begin{aligned}
 \frac{C \prod_{j=1}^n e(C_{j,1}, D_{j,2})}{e(C_1, D_1) \prod_{j=1}^n e(C_{j,2}, D_{j,1})} &= \frac{ck \cdot e(g, g)^{zs} \prod_{j=1}^n e(X_j^{s_j}, g^{vk_j(u-r_i \lambda_j)})}{e(g^s, g^{x+\beta u}) \prod_{j=1}^n e(u_j^{s_j}, X_j^{\lambda_j})} \\
 &= \frac{ck \cdot e(g, g)^{zs} \prod_{j=1}^n e\left(g^{\frac{\beta s_j}{vk_j}}, g^{vk_j(u-r_i \lambda_j)}\right)}{e(g^s, g^{x+\beta u}) \prod_{j=1}^n e\left(g^{-vk_j r_i s_j}, g^{\frac{\beta \lambda_j}{vk_j}}\right)} \\
 &= \frac{ck \cdot e(g, g)^{zs} \prod_{j=1}^n e(g, g)^{\beta s_j u} \prod_{j=1}^n e(g, g)^{-\beta s_j r_i \lambda_j}}{e(g, g)^{sz} e(g, g)^{s\beta u} \prod_{j=1}^n e(g, g)^{-r_i s_j \beta \lambda_j}} \\
 &= \frac{ck \cdot e(g, g)^{\sum_{j=1}^n \beta s_j}}{e(g, g)^{s\beta u}} \\
 &= ck
 \end{aligned}$$

Where $i \in [1, 2n]$.

Attribute revocation: For the sake of achieving attribute revocation, the revocation phase is divided into three steps: Attribute authority takes charge of generating update key, non-revoked data user's secret key update, and cloud server is in charge of updating ciphertext. We assume that the j' -th attribute $attr_{j'}$ of data user will be revoked, where j' may be any one of $1, \dots, n$.

i) The attribute authority takes charge of generating update key

When an attribute needs to be revoked, the attribute authority AA inputs the current version number vk_j of revoked attribute $attr_j$ and chooses a new version number vk_j^* , where $vk_j^* \in \mathbb{Z}_p$ ($vk_j^* \neq vk_j$), then update key generation algorithm calculates the update key as follows:

$$UK_{j'} = \left(UK_{j',1} = \frac{vk_{j'}^*}{vk_{j'}}, UK_{j',2} = \frac{vk_{j'}}{vk_{j'}^*} \right)$$

Afterwards, the attribute authority AA sends $UK_{j'}$ to key generation server KGS and cloud server CS, and updates partial public attribute key that is related to revoked attribute $attr_{j'}$ at

the same time:

$$PK_{j,1}^* = (PK_{j,1})^{UK_{j,1}} = g^{vk_j^*}$$

$$PK_{j,2}^{t*} = (PK_{j,2}^t)^{UK_{j,1}} = (u_j)^{UK_{j,1}} = g^{-vk_j^* r_j}$$

$$PK_{j,2}^{s*} = (PK_{j,2}^s)^{UK_{j,1}} = (u_{j+n})^{UK_{j,1}} = g^{-vk_j^* r_{j+n}}$$

$$PK_{j,3}^{t*} = (PK_{j,3}^t)^{UK_{j,1}} = (y_j)^{UK_{j,1}} = e(x_j, g)^{vk_j^*}$$

$$PK_{j,3}^{s*} = (PK_{j,3}^s)^{UK_{j,1}} = (y_{j+n})^{UK_{j,1}} = e(x_{j+n}, g)^{vk_j^*}$$

$$PK_{j,4}^* = (PK_{j,4})^{UK_{j,2}} = (X_j)^{UK_{j,2}} = g^{\frac{\beta}{vk_j^*}}$$

Consequently, the public attribute key associated with revoked attribute $attr_j$ as

$$PK_j^* = \{PK_{j,1}^*, PK_{j,4}^*, (PK_{j,2}^{t*}, PK_{j,3}^{t*}), (PK_{j,2}^{s*}, PK_{j,3}^{s*})\}$$

At last, this algorithm generates updated public key PK^* and updated master secret key MSK^* :

$$PK^* = (e, \mathbb{G}, \mathbb{G}_T, g, g^a, g^b, g^c, Y, H, PK_j^*, \{PK_j | attr_j \in U, j \neq j'\})$$

$$MSK^* = (a, b, c, \alpha, \beta, \{(r_i, x_i)\}_{i \in [1, 2n]}, MSK_j^* = vk_j^*, \{MSK_j = vk_j | attr_j \in U, j \neq j'\})$$

ii) The non-revoked data user's secret key update

Firstly, attribute authority AA updates the partial intermediate key OK that is related to revoked attribute $attr_{j'}$, this algorithm computes

$$\sigma_{j'}^* = (\sigma_{j'})^{UK_{j',1}} = \begin{cases} (x_{j'} v_{j'}^{r_{j'}})^{vk_{j'}^*}, \hat{v}_{j'} = v_{j'} \\ (x_{j'+n} v_{j'+n}^{r_{j'+n}})^{vk_{j'}^*}, \hat{v}_{j'} = \neg v_{j'} \end{cases}$$

And transmits $\sigma_{j'}^*$ to key generation server KGS.

Secondly, AA updates the partial local secret key SK_2 related to revoked attribute $attr_{j'}$ according to $UK_{j'}$ as follows:

$$SK_2^* = (D_1, D_{j',1}^* = (D_{j',1})^{UK_{j',2}} = g^{\frac{\beta \lambda_{j'}}{vk_{j'}^*}}, D_{j',2}^*, \forall attr_j \in Attr \setminus \{attr_{j'}\} : D_{j,1}, D_{j,2})$$

where

$$D_{j,2}^* = (D_{j,2})^{UK_{j,1}} = \begin{cases} g^{vk_j^*(u-r_{j,1})}, \hat{v}_j = v_j \\ g^{vk_j^*(u-r_{j,1}+n)}, \hat{v}_j = \neg v_j \end{cases}$$

Note that only partial local secret key that is related to revoked attribute $attr_j$ will be updated, others remain unchanged.

Thirdly, KGS updates the partial outsourced secret key that is related to revoked attribute $attr_j$, the algorithm computes:

$$\tilde{\sigma}^* = \prod_{\substack{j=1 \\ j \neq j'}}^n \sigma_j \cdot \sigma_j^*$$

$$y_{j'}^* = (y_{j'})^{UK_{j',1}} = \begin{cases} (y_{j'})^{UK_{j',1}} = e(x_{j'}, g)^{vk_{j'}^*}, \hat{v}_{j'} = v_{j'} \\ (y_{j'+n})^{UK_{j',1}} = e(x_{j'+n}, g)^{vk_{j'}^*}, \hat{v}_{j'} = \neg v_{j'} \end{cases}$$

$$\tilde{y}^* = \prod_{\substack{j=1 \\ j \neq j'}}^n y_j \cdot y_{j'}^*$$

Then, it returns $(\tilde{\sigma}^*, \tilde{y}^*)$ to AA.

Finally, AA sends the updated secret key $SK^* = (SK_1^* = (v, \tilde{\sigma}^*, \tilde{y}^*), SK_2^*)$ to non-revoked data user. In the meanwhile, since token contains the component of secret key, then non-revoked data user's token is updated as

$$Tok^* = (v^z, \tilde{\sigma}^{*z}, \tilde{y}^{*z}, Tok_1, Tok_2).$$

iii) The cloud server is in charge of updating ciphertext

① The update of keyword index

Based on updated key UK_j sent by AA, the cloud server updates partial keyword index that is related to revoked attribute $attr_j$, this algorithm computes:

$$u_j^* = (u_j')^{UK_{j,1}} = \begin{cases} (u_j')^{UK_{j,1}} = g^{-vk_j^* r_j}, \tilde{v}_j = v_j \\ (u_{j+n})^{UK_{j,1}} = g^{-vk_j^* r_{j+n}}, \tilde{v}_j = \neg v_j \end{cases}$$

$$\tilde{u}^* = g^{l_2} \prod_{\substack{j=1 \\ j \neq j'}}^n u_j' \cdot u_{j'}^*$$

Hereafter, the updated keyword index is $Index^* = (\tilde{u}^*, W', W)$.

② The ciphertext update of encryption key

The cloud server CS updates the partial ciphertext of encryption key related to revoked attribute $attr_j$, the updated ciphertext as follows:

$$CT^{**} = (C, C_1, C_{j,1}^{**} = (C_{j,1})^{UK_{j,2}} = g^{\frac{\beta s_j}{vk_j^*}}, C_{j,2}^{**}, \forall attr_j \in Attr \setminus \{attr_j\} : C_{j,1}, C_{j,2})$$

where

$$C_{j,2}^{**} = (C_{j,2})^{UK_{j,1}} = \begin{cases} g^{-vk_j^* r_j s_j}, \tilde{v}_j = v_j \\ g^{-vk_j^* r_j + n s_j}, \tilde{v}_j = \neg v_j \end{cases}$$

Note that only partial ciphertext that is related to revoked attribute $attr_j$ will be updated, others remain unchanged.

Finally, the updated overall ciphertext is $Cph^* = (Index^*, E_{ck}(M), CT^{**})$.

Security proof

5.1 Selectively secure game of chosen-keyword attacks

Theorem 1. If there is an adversary $\mathcal{A}$ who is able to win selectively secure game of chosen-keyword attacks with advantage ε , then advantage of a challenger $\mathcal{C}$ to break the DL assumption can be defined as $\frac{\varepsilon}{2}$.

Proof: Supposing that a tuple $(g, f, h, f^{r_1}, g^{r_2}, R)$ which is an instance of the DL problem, where $g, f, h, R \in \mathbb{G}$, $r_1, r_2 \in \mathbb{Z}_p^*$, is given to a challenger $\mathcal{C}$, the challenger $\mathcal{C}$ performs the following selectively secure game of chosen-keyword attacks with an adversary $\mathcal{A}$, we will proof that if the adversary $\mathcal{A}$ who can break our scheme with advantage ε , then the challenger $\mathcal{C}$ can solve the DL problem with advantage $\frac{\varepsilon}{2}$.

Setup. The challenger $\mathcal{C}$ runs **Setup** algorithm, selects a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}, \mathbb{G}_T$ are two multiplicative cyclic groups with prime order p , g is a generator of group $\mathbb{G}$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ is a one-way hash function. It randomly picks $\{r_1, r_2, \dots, r_{2n}\} \in \mathbb{Z}_p$, $\{x_1, x_2, \dots, x_{2n}\} \in \mathbb{Z}_p$, and implicitly sets $h = g^{r_1}, f = g^{r_2}$, it chooses $d \in \mathbb{Z}_p$ at random, lets $f^d = g^b$, where implies that $b = cd$. For each attribute $attr_j \in U(j \in [1, n])$, the challenger $\mathcal{C}$ chooses $vk_j \in \mathbb{Z}_p$ as initial version number, when $attr_j$ is equal to v_j , we have $u_j = g^{-vk_j r_j}$, when $attr_j$ is equal to $\neg v_j$, we have $u_{j+n} = g^{-vk_j r_j + n}$. Similarly, $\mathcal{C}$ computes $y_j = e(x_j, g)^{vk_j}$ if $attr_j$ is equal to v_j , it computes $y_{j+n} = e(x_{j+n}, g)^{vk_j}$ if $attr_j$ is equal to $\neg v_j$. The public key and master secret key are

$$PK = (e, \mathbb{G}, \mathbb{G}_T, g, h, f^d, f, H, \{(u_j, y_j), (u_{j+n}, y_{j+n})\}_{j \in [1, n]})$$

$$MSK = (a, c, d, \{(r_i, x_i)\}_{i \in [1, 2n]}, \{vk_j\}_{j \in [1, n]})$$

The adversary $\mathcal{A}$ submits a challenge access policy $S^* = \tilde{v}_1 \tilde{v}_2 \dots \tilde{v}_n$ to the challenger $\mathcal{C}$.

Phase 1. The challenger $\mathcal{C}$ initializes an empty keyword set query list L_W , then adversary $\mathcal{A}$ issues the following polynomial times adaptive queries:

Outsourced secret key query: The adversary $\mathcal{A}$ sends a query attribute set $Attr^* = \hat{v}_1 \hat{v}_2 \dots \hat{v}_n$ to challenger $\mathcal{C}$, where $\hat{v}_j = v_j$ or $\hat{v}_j = \neg v_j$.

① If $\gamma(Attr^*, S^*) = 1$, the game outputs termination.

② If $\gamma(Attr^*, S^*) = 0$, the challenger $\mathcal{C}$ computes $v = f^{k'}$, implicitly sets $k' = a$, then it computes

$$\tilde{\sigma} = \prod_{j=1}^n \sigma_j, \tilde{y} = \prod_{j=1}^n y'_j, \text{ where}$$

$$\sigma_j = \begin{cases} (x_j v^{r_j})^{v_{k_j}}, \hat{v}_j = v_j & j \in [1, n] \\ (x_{j+n} v^{r_{j+n}})^{v_{k_j}}, \hat{v}_j = \neg v_j & j \in [1, n] \end{cases}$$

$$y'_j = \begin{cases} y_j, \hat{v}_j = v_j & j \in [1, n] \\ y_{j+n}, \hat{v}_j = \neg v_j & j \in [1, n] \end{cases}$$

Subsequently, the challenger $\mathcal{C}$ transmits $SK_1 = (v, \tilde{\sigma}, \tilde{y})$ to adversary $\mathcal{A}$.

Token query: The adversary $\mathcal{A}$ commits a query keyword set $WD' = (\bar{w}'_1, \bar{w}'_2, \dots, \bar{w}'_d)$. The challenger $\mathcal{C}$ randomly selects $\eta \in \mathbb{Z}_p$ and generates the token of keyword set WD' , that is

$$Tok = \left(Tok_1 = \prod_{t=1}^d (h^{f^{dH(\bar{w}'_t)}})^{\eta}, Tok_2 = f^{\eta}, v^{\eta}, \tilde{\sigma}^{\eta}, \tilde{y}^{\eta} \right)$$

If the query attribute set $Attr^*$ does not satisfy access policy S^* , then the challenger $\mathcal{C}$ adds WD' to list L_W and sends Tok to adversary $\mathcal{A}$.

Challenge. The adversary $\mathcal{A}$ randomly picks two keyword sets $WD'_0 = (\bar{w}'_{0,1}, \bar{w}'_{0,2}, \dots, \bar{w}'_{0,r})$ and $WD'_1 = (\bar{w}'_{1,1}, \bar{w}'_{1,2}, \dots, \bar{w}'_{1,r})$ that are not in list L_W , and forwards to challenger $\mathcal{C}$. The challenger $\mathcal{C}$ throws a fair coin to select $\xi \in \{0, 1\}$, runs **keyword-Encrypt** algorithm, and computes

$$\tilde{u} = g^{t_2} \prod_{j=1}^n u'_j = g^{r_2} \prod_{j=1}^n u'_j$$

$$W' = f^{t_1} = f^{r_1}$$

$$W_t = R f^{dH(\bar{w}'_t, t)^{r_1}}, t \in [1, r]$$

which implies $t_1 = r_1, t_2 = r_2$, where

$$u'_j = \begin{cases} u_j, \tilde{v}_j = v_j & j \in [1, n] \\ u_{j+n}, \tilde{v}_j = \neg v_j & j \in [1, n] \end{cases}$$

As such, the keyword index is $Index = (\tilde{u}, W', \{W_t\}_{t \in [1, r]})$, then the challenger $\mathcal{C}$ returns $Index$ to adversary $\mathcal{A}$.

Phase 2. Except that the keyword set WD'_0 and WD'_1 can no longer be queried, the other queries are similar to Phase 1.

Guess. Finally, the adversary $\mathcal{A}$ makes a guess ξ' of ξ . If $\xi' = \xi$, $\mathcal{A}$ wins game. The challenger $\mathcal{C}$ considers $R = h^{r_1+r_2}$ implicitly. Because when $R = h^{r_1+r_2}$, $Index$ is a legitimate index of keyword set WD'_{ξ} .

Afterwards, we assume that the adversary $\mathcal{A}$ who can break the scheme with non-negligible advantage ϵ , the probability is $\Pr[\xi' = \xi] = \frac{1}{2} + \epsilon$ if $R = h^{r_1+r_2}$, otherwise $\Pr[\xi' = \xi] = \frac{1}{2}$. The

advantage of challenger $\mathcal{C}$ in solving DL assumption is defined as

$$\begin{aligned} ADV(\mathcal{C}) &= \left\{ \frac{1}{2} \Pr[\xi' = \xi | R = h^{r_1+r_2}] + \frac{1}{2} \Pr[\xi' = \xi | R \neq h^{r_1+r_2}] \right\} - \frac{1}{2} \\ &= \frac{1}{2} \times \left(\frac{1}{2} + \epsilon \right) + \frac{1}{2} \times \frac{1}{2} - \frac{1}{2} \\ &= \frac{\epsilon}{2} \end{aligned}$$

Therefore, our scheme will achieve selectively secure against chosen-keyword attacks.

5.2 Token privacy game

Theorem 2. If there is an adversary $\mathcal{A}$ who is able to break token privacy game with a non-negligible advantage, a challenger $\mathcal{C}$ is capable of obtaining w'_t from $H(w'_t)$ with a non-negligible advantage.

Proof: Here we use token privacy game to prove the theorem.

Setup. The challenger $\mathcal{C}$ executes **Setup** algorithm, selects a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}, \mathbb{G}_T$ are two multiplicative cyclic groups with prime order p , g is a generator of group $\mathbb{G}$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ is a one-way hash function. It randomly chooses $a, b, c, \{r_i\}_{i \in [1, 2n]} \leftarrow \mathbb{Z}_p$ and $\{x_i\}_{i \in [1, 2n]} \leftarrow \mathbb{G}$. For each attribute $attr_j \in U$, $vk_j \in \mathbb{Z}_p$ is initial version number, where $j \in [1, n]$. When $attr_j$ is equal to v_j , we have $u_j = g^{-vk_j r_j}$, when $attr_j$ is equal to $\neg v_j$, we have $u_{j+n} = g^{-vk_j r_{j+n}}$. Similarly, $\mathcal{C}$ computes $y_j = e(x_j, g)^{vk_j}$ if $attr_j$ is equal to v_j , it computes $y_{j+n} = e(x_{j+n}, g)^{vk_j}$ if $attr_j$ is equal to $\neg v_j$. The challenger $\mathcal{C}$ generates public key $PK = (e, \mathbb{G}, \mathbb{G}_T, g, g^a, g^b, g^c, H, \{(u_j, y_j), (u_{j+n}, y_{j+n})\}_{j \in [1, n]})$, master secret key $MSK = (a, b, c, \{(r_i, x_i)\}_{i \in [2n]}, \{vk_j\}_{j \in [1, n]})$, while publishing PK and keeping MSK secret.

Phase 1. Similar to selectively secure game of chosen-keyword attacks, the challenger $\mathcal{C}$ initializes an empty key query list L_K , the adversary issues the following polynomial q_t times adaptive queries:

Outsourced secret key query: The adversary $\mathcal{A}$ selects a query attribute set $Attr = \hat{v}_1 \hat{v}_2 \cdots \hat{v}_n$ as input for **In-KeyGen** algorithm, where $\hat{v}_j = v_j$ or $\hat{v}_j = \neg v_j$, the challenger $\mathcal{C}$ computes $v = g^{ac}$ and

$$\sigma_j = \begin{cases} (x_j v_j)^{vk_j}, & \hat{v}_j = v_j \\ (x_{j+n} v_{j+n})^{vk_j}, & \hat{v}_j = \neg v_j \end{cases} \quad j \in [1, n]$$

Hereafter, it obtains the intermediate key

$$OK = (Attr, v, \{\sigma_j\}_{j \in [1, n]})$$

The challenger $\mathcal{C}$ runs **Out-KeyGen** algorithm to output the outsourced secret key

$$SK_1 = \left(v, \tilde{\sigma} = \prod_{j=1}^n \sigma_j, \tilde{y} = \prod_{j=1}^n y'_j \right), \text{ where } y'_j = \begin{cases} y_j, & \hat{v}_j = v_j \\ y_{j+n}, & \hat{v}_j = \neg v_j \end{cases} \quad j \in [1, n]$$

Then, the challenger $\mathcal{C}$ transmits SK_1 to adversary $\mathcal{A}$, and adds $Attr$ to list L_K .

Token query: The challenger $\mathcal{C}$ runs **TokenGen** algorithm based on outsourced secret key SK_1 and keyword set $WD' = (w'_1, w'_2, \dots, w'_d)$ submitted by adversary $\mathcal{A}$, then it randomly

picks $z' \in \mathbb{Z}_p$ to compute $Tok_1 = \prod_{t=1}^d (g^a g^{bH(w'_t)})^{z'}$ and $Tok_2 = g^{cz'}$, sends the token $Tok = (v^{z'}, \tilde{\sigma}^{z'}, \tilde{y}^{z'}, Tok_1, Tok_2)$ to adversary $\mathcal{A}$.

Challenge. The adversary $\mathcal{A}$ submits an access policy S^* with the restriction that the attribute set $Attr$ in list L_K does not satisfy access policy S^* , namely $\gamma(Attr, S^*) = 0$. The challenger $\mathcal{C}$ randomly selects a keyword set WD^* , encrypts WD^* with S^* to obtain $Index^*$, then it chooses an attribute set $Attr^*$ such that $Attr^*$ satisfies access policy S^* , that is $\gamma(Attr^*, S^*) = 1$. Subsequently, the challenger $\mathcal{C}$ first executes **In-KeyGen** algorithm and **Out-KeyGen** algorithm to compute SK_1^* , then runs **TokenGen** algorithm to gain Tok^* , and sends Tok^* to adversary $\mathcal{A}$.

Phase 2. Similar to Phase 1, but keyword set WD^* can no longer be queried.

Guess. The adversary $\mathcal{A}$ outputs a keyword set WD'' and transmits to challenger $\mathcal{C}$. If $WD'' = WD^*$, adversary $\mathcal{A}$ wins game. In other words, the challenger $\mathcal{C}$ performs **Keyword-Encrypt** algorithm to generate the ciphertext $Index''$ of WD'' , if the result returned by **Search**($Index''$, Tok^*) algorithm is 1, then the adversary $\mathcal{A}$ wins game.

Particularly, if adversary $\mathcal{A}$ attempts to gain keyword information from the index, it needs

to analyze $Tok_1 = \prod_{t=1}^d (g^a g^{bH(w'_t)})^{z'}$, where z' is randomly selected by challenger $\mathcal{C}$, thus the adversary $\mathcal{A}$ gets information from $H(w'_t)$ at most. Due to the unidirectional and collision-resistant properties of the hash function, if adversary $\mathcal{A}$ obtains the advantage of w'_t from $H(w'_t)$ is negligible σ , $|\Pi|$ represents the size of keyword space, q_t represents the number of inquiries of Phases 1, $|\Pi| - q_t$ is the size of the remaining keyword set in the keyword space after the phase 1 inquiry. The probability that the adversary $\mathcal{A}$ guesses the encrypted keyword from the remaining keyword set is $\frac{1}{|\Pi| - q_t}$. Then, the advantage of adversary $\mathcal{A}$ to break token privacy game can be defined at most $ADV(\mathcal{A}) = \frac{1}{|\Pi| - q_t} + \sigma$. In practical applications, because $|\Pi|$ is large enough and q_t is finite size, then the advantage $ADV(\mathcal{A}) = \frac{1}{|\Pi| - q_t} + \sigma$ is negligible, therefore our proposal can guarantee token privacy security.

5.3 Selectively secure game of chosen-plaintext attacks

Our scheme is selective security under the assumption. For selective security, the adversary $\mathcal{A}$ should submit two challenge access policies S_0^* and S_1^* at the beginning of game, then the scheme is to prove that the adversary $\mathcal{A}$ cannot win the game with a non-negligible advantage through the indistinguishability of a series of games. In simple terms, we use $(C^*, C_1^*, \{(C_{j,1}^*, C_{j,2}^*)\}_{j \in [1, n]})$ to denote initial challenge ciphertext, then select random element Z in $\mathbb{G}_T$ and $\{Z_{j,1}\}_{j \in [1, n]}$ in $\mathbb{G}$. Afterwards, we define a series of games $Game_0, \dots, Game_{n+1}$, where $Game_0$ is initial game, $Game_1$ is to replace C^* in $Game_0$ with Z , $Game_2, \dots, Game_{n+1}$ are to replace $C_{j,1}^*$ with $Z_{j,1}$, for $j \in [1, n]$. The game is defined as follows:

$Game_0$: The challenge ciphertext is $CT_0^* = (C^*, C_1^*, \{(C_{j,1}^*, C_{j,2}^*)\}_{j \in [1, n]})$

$Game_1$: The challenge ciphertext is $CT_1^* = (Z, C_1^*, \{(C_{j,1}^*, C_{j,2}^*)\}_{j \in [1, n]})$

$Game_2$: The challenge ciphertext is $CT_2^* = (Z, C_1^*, (Z_{1,1}, C_{1,2}^*), \{(C_{j,1}^*, C_{j,2}^*)\}_{j \in [2, n]})$

.....

$Game_{n+1}$: The challenge ciphertext is $CT_{n+1}^* = (Z, C_1^*, \{(Z_{j,1}, C_{j,2}^*)\}_{j \in [1, n]})$

When a data user's attribute set $Attr$ satisfies both access structures S_0^* and S_1^* , namely $\gamma(Attr, S_0^*) = 1 \wedge \gamma(Attr, S_1^*) = 1$, $Game_1$ is the same as $Game_0$. When a data user's attribute set $Attr$ does not satisfy access policies S_0^* and S_1^* , that is $\gamma(Attr, S_0^*) = 0 \wedge \gamma(Attr, S_1^*) = 0$, $Game_1$ is as defined above. Subsequently, we use the DDH assumption to prove that $Game_0$ and $Game_1$ are indistinguishable.

Theorem 3. If there is an adversary $\mathcal{A}$ who is able to distinguish games $Game_0$ and $Game_1$ with a non-negligible advantage, a simulator $\mathcal{B}$ is capable of breaking DDH assumption with a non-negligible advantage.

Proof: Supposing that a tuple (g, g^{z_1}, g^{z_2}, Q) which is an instance of DDH problem, where $g, Q \in \mathbb{G}$, $z_1, z_2 \in \mathbb{Z}_p^*$, is given to a simulator $\mathcal{B}$, the simulator $\mathcal{B}$ performs the following operations:

Initialization. The adversary $\mathcal{A}$ submits two challenge access policies $S_0^* = \tilde{v}_{0,1} \tilde{v}_{0,2} \cdots \tilde{v}_{0,n}$ and $S_1^* = \tilde{v}_{1,1} \tilde{v}_{1,2} \cdots \tilde{v}_{1,n}$ to simulator $\mathcal{B}$, then $\mathcal{B}$ throws a fair coin to select $b \in \{0,1\}$.

Setup. The simulator $\mathcal{B}$ runs **Setup** algorithm, selects a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}, \mathbb{G}_T$ are two multiplicative cyclic groups with prime order p , g is a generator of group $\mathbb{G}$. It randomly chooses $\alpha, \beta, vk_j \in \mathbb{Z}_p$, sets $Y = e(g, g)^\alpha = e(g, g^{z_1}) = e(g, g)^{z_1}$, implicitly lets $\alpha = z_1$, then it computes $X_j = g^{\frac{\beta}{vk_j}}$, where $vk_j \in \mathbb{Z}_p$ as initial version number of attribute, for $j \in [1, n]$. Afterwards, $\mathcal{B}$ picks up random value $\{r_i \in \mathbb{Z}_p\}_{i \in [1, 2n]}$, when $\hat{v}_j \in S_b^*$, there are $u_j = g^{-vk_j r_j} (\hat{v}_j = v_j = \tilde{v}_j)$ and $u_{j+n} = g^{-vk_j r_{j+n}} (\hat{v}_j = \neg v_j = \tilde{v}_j)$, when $\hat{v}_j \notin S_b^*$, there are $\bar{u}_j = g^{-z_1 vk_j r_j} (\hat{v}_j = v_j \neq \tilde{v}_j)$ and $\bar{u}_{j+n} = g^{-z_1 vk_j r_{j+n}} (\hat{v}_j = \neg v_j \neq \tilde{v}_j)$, where $j \in [1, n]$. The simulator $\mathcal{B}$ publishes public key

$$PK = (e, \mathbb{G}, \mathbb{G}_T, g, Y, \{(X_j, u_j, u_{j+n}, \bar{u}_j, \bar{u}_{j+n})\}_{j \in [1, n]})$$

and keeps master secret key $MSK = (\alpha, \beta, \{vk_j\}_{j \in [1, n]}, \{r_i\}_{i \in [1, 2n]})$ secret.

Phase 1. The adversary $\mathcal{A}$ commits a query attribute set $Attr = \hat{v}_1 \hat{v}_2 \cdots \hat{v}_n$ to simulator $\mathcal{B}$.

Secret key query: Considering that $\gamma(Attr, S_0^*) = 0 \wedge \gamma(Attr, S_1^*) = 0$, thus there is $j'' \in \{1, 2, \dots, n\}$ such that $\hat{v}_{j''} \notin S_b^*$. Then, the simulator $\mathcal{B}$ picks up $u, \lambda_j \in \mathbb{Z}_p$ at random, D_1 and $D_{j,1}$ in the local secret key SK_2 are calculated as $D_1 = g^{\alpha+\beta u} = g^{z_1+\beta u}$ and $D_{j,1} = X_j^{\lambda_j} = g^{\frac{\beta \lambda_j}{vk_j}}$, where $j \in [1, n]$. For $j = j''$, $\mathcal{B}$ computes

$$D_{j,2} = \begin{cases} \bar{u}_j^{\lambda_j} g^{vk_j u} = (g^{-z_1})^{vk_j r_j \lambda_j} g^{vk_j u}, & \hat{v}_j = v_j \\ \bar{u}_{j+n}^{\lambda_j} g^{vk_j u} = (g^{-z_1})^{vk_j r_{j+n} \lambda_j} g^{vk_j u}, & \hat{v}_j = \neg v_j \end{cases}$$

for $j \neq j''$, $\mathcal{B}$ computes

$$D_{j,2} = \begin{cases} u_j^{\lambda_j} g^{vk_j u} = g^{-vk_j r_j \lambda_j} g^{vk_j u}, & \hat{v}_j = v_j \\ u_{j+n}^{\lambda_j} g^{vk_j u} = g^{-vk_j r_{j+n} \lambda_j} g^{vk_j u}, & \hat{v}_j = \neg v_j \end{cases}$$

Afterwards, the simulator $\mathcal{B}$ returns $SK_2 = (D_1, \{(D_{j,1}, D_{j,2})\}_{j \in [1, n]})$ to adversary $\mathcal{A}$.

Challenge. For access policies $S_0^* = \tilde{v}_{0,1} \tilde{v}_{0,2} \cdots \tilde{v}_{0,n}$ and $S_1^* = \tilde{v}_{1,1} \tilde{v}_{1,2} \cdots \tilde{v}_{1,n}$, the adversary $\mathcal{A}$ submits two equal-length encryption keys ck_1 and ck_2 . Then, the simulator $\mathcal{B}$ executes **Encryption key-Encrypt** algorithm with S_b^* and ck_b , where $b \in \{0,1\}$. It sets $C_1^* = g^s = g^{z_2}$, $C^* = ck_b \cdot Y^s = ck_b \cdot e(g, g)^{\alpha s} = ck_b \cdot e(g, Q)$, implicitly lets $s = z_2$. For all $j \in [1, n]$, the simulator $\mathcal{B}$ arbitrarily chooses

$$s_j \in \mathbb{Z}_p \text{ if } j \neq j'', \text{ it computes } s_{j'} = z_2 - \sum_{j=1, j \neq j''}^n s_j \text{ if } j = j''.$$

When $j \neq j''$, $[C_{j',1}^*, C_{j',2}^*]$ in the ciphertext are computed as

$$C_{j',1}^* = X_{j'}^{s_{j'}} = g^{\frac{\beta}{vk_{j'}} \left(z_2 - \sum_{j=1, j \neq j'}^n s_j \right)} = (g^{z_2})^{\frac{\beta}{vk_{j'}}} / g^{\frac{\beta}{vk_{j'}} \sum_{j=1, j \neq j'}^n s_j}$$

$$C_{j',2}^* = u_{j'}^{s_{j'}} = \begin{cases} \bar{u}_{j'}^{s_{j'}} = g^{-z_1 vk_{j'} r_{j'}} \left(z_2 - \sum_{j=1, j \neq j'}^n s_j \right), & \tilde{v}_{bj'} = \neg v_{j'} \\ \bar{u}_{j'+n}^{s_{j'}} = g^{-z_1 vk_{j'} r_{j'} + n} \left(z_2 - \sum_{j=1, j \neq j'}^n s_j \right), & \tilde{v}_{bj'} = v_{j'} \end{cases}$$

When $j \neq j''$, $[C_{j,1}^*, C_{j,2}^*]$ in the ciphertext is computed as

$$C_{j,1}^* = X_j^{s_j} = g^{\frac{\beta s_j}{vk_j}}$$

$$C_{j,2}^* = u_j^{s_j} = \begin{cases} u_j^{s_j} = g^{-vk_j r_j s_j}, & \tilde{v}_{bj} = v_j \\ u_{j+n}^{s_j} = g^{-vk_j r_{j+n} s_j}, & \tilde{v}_{bj} = \neg v_j \end{cases}$$

Finally, the simulator $\mathcal{B}$ sends $CT^{**} = (C^*, C_1^*, \{(C_{j,1}^*, C_{j,2}^*)\}_{j \in [1,n]})$ to adversary $\mathcal{A}$.

Phase 2. The adversary $\mathcal{A}$ makes queries similarly to Phase 1 with the restriction that $\gamma(Attr, S_0^*) = 0 \wedge \gamma(Attr, S_1^*) = 0$.

Guess. The adversary $\mathcal{A}$ gives a guess b' of b . If $b' = b$, the simulator $\mathcal{B}$ outputs $\tau = 1$, it demonstrates $Q = g^{z_1 z_2}$. The adversary $\mathcal{A}$ exactly simulates the initial game, because

$$C^* = ck_b \cdot Y^s = ck_b \cdot e(g, g)^{zs} = ck_b \cdot e(g, Q) = ck_b \cdot e(g, g)^{z_1 z_2}$$

Otherwise the simulator $\mathcal{B}$ outputs $\tau = 0$, it demonstrates that Q is a random element in $\mathbb{G}$.

The advantage of adversary $\mathcal{A}$ to win the game is defined as

$$ADV(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

Hence, if adversary $\mathcal{A}$ who is able to distinguish games $Game_0$ and $Game_1$ with a non-negligible advantage, simulator $\mathcal{B}$ is capable of breaking DDH assumption with a non-negligible advantage.

If $\hat{v}_j$ belongs to $(\hat{v}_j \in S_0^* \wedge \hat{v}_j \in S_1^*)$ or $(\hat{v}_j \notin S_0^* \wedge \hat{v}_j \notin S_1^*)$, the ciphertext component $\{C_{j,1}\}_{j \in [1,n]}$ is the same as initial scheme, but for $\hat{v}_j$ belongs to $(\hat{v}_j \in S_0^* \wedge \hat{v}_j \notin S_1^*)$ or $(\hat{v}_j \notin S_0^* \wedge \hat{v}_j \in S_1^*)$, the ciphertext component $\{C_{j,1}\}_{j \in [1,n]}$ will be replaced by random value $\{Z_{j,1}\}_{j \in [1,n]}$, which is represented as our pre-defined a series of games $Game_2, \dots, Game_n, Game_{n+1}$. The DDH assumption is used to prove that $Game_l$ and $Game_{l+1}$ are indistinguishable, for $l \in [1, n]$.

Theorem 4. If there is an adversary $\mathcal{A}$ who is able to distinguish games $Game_l$ and $Game_{l+1}$ with a non-negligible advantage, a simulator $\mathcal{B}$ is capable of breaking DDH assumption with a non-negligible advantage.

Proof: Supposing that a tuple (g, g^{z_1}, g^{z_2}, Q) which is an instance of DDH problem, where $g, Q \in \mathbb{G}$, $z_1, z_2 \in \mathbb{Z}_p^*$, is given to a simulator $\mathcal{B}$, the simulator $\mathcal{B}$ performs the following operations:

Initialization. The adversary $\mathcal{A}$ submits two challenge access policies $S_0^* = \tilde{v}_{0,1} \tilde{v}_{0,2} \cdots \tilde{v}_{0,n}$ and $S_1^* = \tilde{v}_{1,1} \tilde{v}_{1,2} \cdots \tilde{v}_{1,n}$ to simulator $\mathcal{B}$, then $\mathcal{B}$ throws a fair coin to select $b \in \{0, 1\}$. When an

attribute set $Attr = \hat{v}_1 \hat{v}_2 \cdots \hat{v}_n$ satisfies that $(\hat{v}_j \in S_0^* \wedge \hat{v}_j \in S_1^*)$ or $(\hat{v}_j \notin S_0^* \wedge \hat{v}_j \notin S_1^*)$, $Game_l$ and $Game_{l+1}$ are the same according to game definition. Therefore, only the case of $(\hat{v}_j \in S_0^* \wedge \hat{v}_j \notin S_1^*)$ or $(\hat{v}_j \notin S_0^* \wedge \hat{v}_j \in S_1^*)$ is considered below.

Setup. The simulator $\mathcal{B}$ runs **Setup** algorithm, selects a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}, \mathbb{G}_T$ are two multiplicative cyclic groups with prime order p , g is a generator of group $\mathbb{G}$. It randomly chooses $\alpha, \beta, vk_j \in \mathbb{Z}_p$, sets $Y = e(g, g)^\alpha$, it computes $X_j = g^{\frac{\beta}{vk_j}} = g^{\frac{z_1}{vk_j}}$, where $vk_j \in \mathbb{Z}_p (j \in [1, n])$ as initial version number of attribute, implicitly lets $\beta = z_1$. Afterwards, $\mathcal{B}$ picks up $\{r_i \in \mathbb{Z}_p\}_{i \in [1, 2n]}$ at random, when $\hat{v}_j \in S_b^*$, there are $u_j = g^{-vk_j r_j} (\hat{v}_j = v_j = \tilde{v}_j)$ and $u_{j+n} = g^{-vk_j r_{j+n}} (\hat{v}_j = \neg v_j = \tilde{v}_j)$, when $\hat{v}_j \notin S_b^*$, there are $\bar{u}_j = g^{-z_1 vk_j r_j} (\hat{v}_j = v_j \neq \tilde{v}_j)$ and $\bar{u}_{j+n} = g^{-z_1 vk_j r_{j+n}} (\hat{v}_j = \neg v_j \neq \tilde{v}_j)$, where $j \in [1, n]$. The simulator $\mathcal{B}$ publishes public key $PK = (e, \mathbb{G}, \mathbb{G}_T, g, Y, \{(X_j, u_j, u_{j+n}, \bar{u}_j, \bar{u}_{j+n})\}_{j \in [1, n]})$ and keeps master secret key $MSK = (\alpha, \beta, \{vk_j\}_{j \in [1, n]}, \{r_i\}_{i \in [1, 2n]})$ secret.

Phase 1. The adversary $\mathcal{A}$ commits a query attribute set $Attr = \hat{v}_1 \hat{v}_2 \cdots \hat{v}_n$ to simulator $\mathcal{B}$.

Secret key query: The simulator $\mathcal{B}$ randomly chooses $u, \lambda_j \in \mathbb{Z}_p, j \in [1, n]$. The local secret key SK_2 is computed as $D_1 = g^{\alpha + \beta u} = g^{\alpha + z_1 u}$ and $D_{j,1} = X_j^{\lambda_j} = g^{\frac{\beta \lambda_j}{vk_j}} = g^{\frac{z_1 \lambda_j}{vk_j}}$, when $\hat{v}_j = v_j$, $\mathcal{B}$ computes $D_{j,2} = g^{vk_j(u - r_j \lambda_j)}$, otherwise $D_{j,2} = g^{vk_j(u - r_{j+n} \lambda_j)}$, where $j \in [1, n]$.

The simulator $\mathcal{B}$ sends $SK_2 = (D_1, \{(D_{j,1}, D_{j,2})\}_{j \in [1, n]})$ to adversary $\mathcal{A}$.

Challenge. For access policies $S_0^* = \tilde{v}_{0,1} \tilde{v}_{0,2} \cdots \tilde{v}_{0,n}$ and $S_1^* = \tilde{v}_{1,1} \tilde{v}_{1,2} \cdots \tilde{v}_{1,n}$, the adversary $\mathcal{A}$ submits two equal-length keys ck_1 and ck_2 . The simulator $\mathcal{B}$ executes **Encryption key-Encrypt** algorithm with S_b^* and ck_b , where $b \in \{0, 1\}$. It sets $C_1^* = g^s = g^{z_2}$ and $C^* = ck_b \cdot Y^s = ck_b \cdot e(g, g)^{z_2} = ck_b \cdot e(g, g^{z_2})^\alpha$, implicitly lets $s = z_2$. Hereafter, for all $j \in [1, n]$, the simulator $\mathcal{B}$ arbitrarily chooses $s_j \in \mathbb{Z}_p$ if $j \neq n$, it computes $s_n = z_2 - \sum_{j=1, j \neq n}^n s_j$ if $j = n$.

When $j = n$, $[C_{n,1}^*, C_{n,2}^*]$ in the ciphertext is computed as

$$C_{n,1}^* = X_n^{s_n} = g^{\frac{z_1}{vk_n} \left(z_2 - \sum_{j=1, j \neq n}^n s_j \right)} = Q_n^{\frac{1}{vk_n}} / g^{\frac{z_1}{vk_n} \sum_{j=1, j \neq n}^n s_j}$$

$$C_{n,2}^* = u_n'^{s_n} = \begin{cases} u_n^{s_n} = g^{-vk_n r_n \left(z_2 - \sum_{j=1, j \neq n}^n s_j \right)}, & \tilde{v}_{b,n} = v_n \\ u_{2n}^n = g^{-vk_n r_{2n} \left(z_2 - \sum_{j=1, j \neq n}^n s_j \right)}, & \tilde{v}_{b,n} = \neg v_n \end{cases}$$

When $j \neq n$, $[C_{j,1}^*, C_{j,2}^*]$ in the ciphertext is computed as

$$C_{j,1}^* = X_j^{s_j} = g^{\frac{\beta s_j}{vk_j}}$$

$$C_{j,2}^* = u_j'^{s_j} = \begin{cases} \bar{u}_j^{s_j} = g^{-z_1 vk_j r_j s_j}, & \tilde{v}_{b,j} = \neg v_j \\ \bar{u}_{j+n}^{s_j} = g^{-z_1 vk_j r_{j+n} s_j}, & \tilde{v}_{b,j} = v_j \end{cases}$$

Then the simulator $\mathcal{B}$ transmits $CT^{**} = (C^*, C_1^*, \{(C_{j,1}^*, C_{j,2}^*)\}_{j \in [1, n]})$ to adversary $\mathcal{A}$.

Phase 2. The adversary $\mathcal{A}$ continues to query similarly to Phase 1 with the restriction that $(\hat{v}_j \in S_0^* \wedge \hat{v}_j \notin S_1^*)$ or $(\hat{v}_j \notin S_0^* \wedge \hat{v}_j \in S_1^*)$.

Guess. The adversary $\mathcal{A}$ gives a guess b' of b . If $b' = b$, the simulator $\mathcal{B}$ outputs $\tau = 1$, it demonstrates $Q = g^{z_1 z_2}$. The adversary $\mathcal{A}$ exactly simulates the initial game, because

$$C_{n,1}^* = X_n^{s_1} = g^{\frac{z_1}{vk_n} \left(z_2 - \sum_{j=1, j \neq n}^n s_j \right)} = Q^{\frac{1}{vk_n}} / g^{\frac{z_1}{vk_n} \sum_{j=1, j \neq n}^n s_j} = (g^{z_1 z_2})^{\frac{1}{vk_n}} / g^{\frac{z_1}{vk_n} \sum_{j=1, j \neq n}^n s_j}$$

Otherwise the simulator $\mathcal{B}$ outputs $\tau = 0$, it demonstrates that Q is a random element in $\mathbb{G}$. The advantage of adversary $\mathcal{A}$ to win the game is defined as

$$ADV(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

Therefore, if adversary $\mathcal{A}$ who is able to distinguish games $Game_i$ and $Game_{i+1}$ with a non-negligible advantage, simulator $\mathcal{B}$ is capable of breaking DDH assumption with a non-negligible advantage.

In a word, if for all polynomial-time adversary who could win the game with a negligible advantage, our scheme will achieve selective ciphertext security under chosen-plaintext attacks.

5.4 Forward and backward security

A data user's attribute set satisfies access structure, only when the search keyword set is included in the encrypted keyword set, the target ciphertext can be searched. However, an attribute $attr_j$ of data user needs to be revoked at a certain time, the system will inform all non-revoked data users to update secret keys that associated with revoked attribute $attr_j$ and the cloud server completes ciphertexts update. During the initial phase, we select a randomized version number for each attribute. As long as update occurs, the version number of attribute will be changed. Afterwards, an update key UK_j will be used to re-encrypt ciphertexts, and update secret keys relevant to revoked attribute of all non-revoked data users, the data user cannot continue performing keyword search and decryption with previous token and secret key. Because in the search phase, the updated keyword index cannot match data user's original token, namely

$$\frac{e(\tilde{u}^*, v^z) e(\tilde{\sigma}^z, g)}{\tilde{y}^z} \neq e(g, g)^{t_2 a c z} \quad (vk_j^* \neq vk_j)$$

During the decryption phase, the data user's original secret key cannot decrypt updated ciphertext, that is

$$\frac{C \prod_{j=1, j \neq j'}^n e(C_{j,1}, D_{j,2}) e(C_{j',1}^*, D_{j',2})}{e(C_1, D_1) \prod_{j=1, j \neq j'}^n e(C_{j,2}, D_{j,1}) e(C_{j',2}^*, D_{j',1})} \neq ck \quad (vk_j^* \neq vk_j)$$

Therefore, the attribute revocation in our scheme achieves backward security.

A data user's attribute set satisfies access structure, when an attribute $attr_j$ of data user needs to be revoked, the version number vk_j corresponding to attribute will be changed, namely it will be randomized by a random value vk_j^* . Hereafter, the system will inform update

secret keys that related to revoked attribute of all non-revoked data users, updated secret keys and tokens are returned to effective data users. Simultaneously, the partial ciphertexts are relevant to revoked attribute are also updated. Even if the data user saves the ciphertext before joining system, because of $vk_j^* \neq vk_j$, the previous keyword cannot be searched with the updated token and the previous ciphertext cannot be decrypted with the updated secret key. Therefore, the attribute revocation in our scheme achieves forward security.

Performance and efficiency analyses

This section mainly analyzes performance and efficiency comparisons between our scheme and literature [31–35]. The performance comparison is to compare functional differences between our scheme and other schemes. The efficiency comparison is to analyze operation time differences both our proposal and other schemes in a certain phase. Based on Pairing Cryptography (PBC) library [36] and group operation with prime order p , we mainly consider three kinds of operations on time complexity: exponential operation, multiplication operation and pair operation. Specifically, K represents the number of attributes that satisfy access structure and N represents the number of attributes owned by data user. (In our scheme, $K = N$)

Table 1 shows performance comparison between our scheme and literature [31–35]. The literature [31] supports multi-keyword search and security proof under the hard problem, but it does not support attribute-based encryption and revocation. The literature [32–35] all support attribute-based encryption, meanwhile literature [35] also supports direct revocation, while our scheme supports attribute revocation. In addition, the literature [32] does not support security proof under the hard problem. Compared with other solutions in the Table 1, our proposal supports all functions.

Table 2 shows efficiency comparison between the proposed scheme and the literature [31–35]. Since literature [32–35] do not support multi-keyword search, the proposed scheme and literature [31] have requirements for the number of keywords in the encryption, token generation, and search phases. For comparison, we consider that the quantity of encrypted keywords is equivalent to the quantity of search keywords, that is $r = d$.

Through experiments on common cryptographic algorithms, the operation time of each phase is obtained and the following operation time comparison chart is drawn. The environment of the hardware runtime is Intel Core i5-3470 CPU @ 3.20GHz, and RAM is 4.00GB. The software runtime environment is JDK 1.7.5, JPBC 2.0.0 and MyEclipse 10.

In Fig 2(A), 2(B) and 2(C) are time charts for running encryption algorithm as the number of attributes increases when keyword set contains 10, 20, and 30 keywords, respectively. Since literature [32–35] do not support multi-keyword search function, the proposed scheme is

Table 1. Performance comparison.

Scheme	Multi-keyword search	Attribute-based encryption	revocation	Security proof under the hard problem
[31]	✓	×	×	✓
[32]	×	✓	×	×
[33]	×	✓	×	✓
[34]	×	✓	×	✓
[35]	×	✓	✓	✓
Ours	✓	✓	✓	✓

×: The scheme does not support corresponding function

✓: The scheme supports corresponding function

<https://doi.org/10.1371/journal.pone.0205675.t001>

Table 2. Efficiency comparison.

Scheme	Encryption	Token generation	Search	Decryption
[31]	$(2K+2r+1)E+(K+r)M'+(2r+1)E_T$	$(4N+2d)E+(2N+d)M'$	$(2N+d)P+dE_T+M_T$	—
[32]	$KP+(4K+1)E+KM'+(2K+1)E_T+KM_T$	—	—	$3NP+(3N+1)M_T$
[33]	$(2K+3)E+(K+1)M'+E_T$	$(4N+5)E+(2N+2)M'$	$3P+E_T+2M_T$	$(2N+2)P+(N+1)M'+2NE_T+3M_T$
[34]	$(8K+3)E+E_T+(2K+1)M'$	—	—	$2E+M'+NE_T+(5N+1)M_T+(6N+1)P$
[35]	$(4K+2)E+(3K-1)M'+E_T$	—	—	$(N-1)E+(N-2)M'+(N+1)E_T+(2N+4)M_T+(2N+4)P$
Ours	$(2K+r+4)E+(K+r)M'+E_T$	$(2N+d+4)E+(N+d)M'+E_T+NM_T$	$(d+3)P+(d+3)M_T$	$(2N+1)P+(2N+1)M_T$

E : An exponential operation of elements in group $\mathbb{G}$

E_T : An exponential operation of elements in group $\mathbb{G}_T$

M' : A multiplication operation on group $\mathbb{G}$

M_T : A multiplication operation on group $\mathbb{G}_T$

P : A pair operation

—: The scheme does not include this calculation

<https://doi.org/10.1371/journal.pone.0205675.t002>

mainly compared with literature [31]. From the Fig 2, we can see that in the encryption phase, the calculation speed of our scheme is slower than literature [33], faster than literature [31,32,34,35].

In Fig 3(D), 3(E) and 3(F) are time charts for running token generation algorithm as the number of attributes increases when keyword set contains 10, 20, and 30 keywords, respectively. Since literature [32,34,35] does not involve token generation algorithm, at this phase, our scheme is mainly compared with literature [31,33], and because literature [33] does not support multi-keyword search function, the changes in the number of keywords will only lead to changes in the algorithm time of our scheme and literature [31]. As can be seen from the Fig 3, when the number of keywords is 10, the calculation time of our scheme is shortest. When the number of keywords increases and the number of attributes is small, the computation speed of the proposed scheme is slower than literature [33]. However, as the number of attributes increases, the advantage of our scheme become more significant, and the algorithm speed is superior to literature [31,33].

In Fig 4(G), 4(H) and 4(I) are the time charts for running search algorithm as the number of attributes increases when keyword set contains 10, 20, and 30 keywords, respectively. Since literature [33] does not involve the number of attributes and keywords in the search phase, the algorithm time in [33] is constant. We have focused on comparing our scheme with literature [31], Fig 4(H) and Fig 4(I) no longer calculate the computation time of literature [33]. Since our scheme do not involve the number of attributes in the search phase, the algorithm time of our scheme is only associated with the number of keywords. From the Fig 4, we can see that at this phase, the calculation speed of the proposed scheme is faster than literature [31].

The Fig 5 is the algorithm time figure in the decryption phase of our scheme and literature [32–35]. It can be seen from the Fig 5 that with continuous increase of the quantity of attributes, calculation time of our solution in this phase is less than literature [32–35].

Conclusion

We propose an attribute-based encryption scheme with multi-keyword search and supporting attribute revocation in cloud storage. On the one hand, based on the traditional CP-ABE schemes, our proposal uses the AND-gate access policy, adds attribute revocation and multi-keyword search, which is more practical. On the other hand, we present security proof under the complexity assumption, it demonstrates that our scheme is secure. Finally, we analyze

(a) Encryption time for 10 keywords

(b) Encryption time for 20 keywords

(c) Encryption time for 30 keywords

Fig 2. (a) Encryption time for 10 keywords (b) Encryption time for 20 keywords. (c) Encryption time for 30 keywords.

<https://doi.org/10.1371/journal.pone.0205675.g002>

performance and efficiency of proposed scheme and other solutions via experimental evaluation, which indicates that our scheme is more efficient.

In the future, how to achieve a dynamic search with multiple functions and direct revocation will be a project that needs further study.

(d) Token generation time for 10 keywords

(e) Token generation time for 20 keywords

(f) Token generation time for 30 keywords

Fig 3. (d) Token generation time for 10 keywords (e) Token generation time for 20 keywords (f) Token generation time for 30 keywords.

<https://doi.org/10.1371/journal.pone.0205675.g003>

(g) Search time for 10 keywords

(h) Search time for 20 keywords

(i) Search time for 30 keywords

Fig 4. (g) Search time for 10 keywords (h) Search time for 20 keywords (i) Search time for 30 keywords.

<https://doi.org/10.1371/journal.pone.0205675.g004>

Fig 5. Decryption time.

<https://doi.org/10.1371/journal.pone.0205675.g005>

Supporting information

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

S2 File. Notation definition.
(DOCX)

Acknowledgments

Thanks to the anonymous reviewer for their useful comments.

Author Contributions

Formal analysis: Shangping Wang.

Methodology: Shangping Wang.

Writing – original draft: Lisha Yao.

Writing – review & editing: Yaling Zhang.

References

1. Mell PM, Grance T (2011) SP 800–145. The NIST Definition of Cloud Computing: National Institute of Standards & Technology. 50–50 p.
2. Sahai A, Waters B. Fuzzy Identity-Based Encryption; 2005. pp. 457–473.
3. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data; 2006. pp. 89–98.
4. Shi Y, Zheng Q, Liu J, Han Z (2015) Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation. *Information Sciences An International Journal* 295: 221–231.
5. Rahulamathavan Y, Veluru S, Han J, Li F, Rajarajan M, Lu R (2016) User Collusion Avoidance Scheme for Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption. *IEEE Transactions on Computers* 65: 2939–2946.
6. Meng R, Zhou Y, Ning J, Liang K, Han J, Susilo W. An Efficient Key-Policy Attribute-Based Searchable Encryption in Prime-Order Groups; 2017. pp. 39–56.
7. Bethencourt J, Sahai A, Waters B. Ciphertext-Policy Attribute-Based Encryption; 2007. pp. 321–334.
8. Goyal V, Jain A, Pandey O, Sahai A. Bounded Ciphertext Policy Attribute Based Encryption; 2008. pp. 579–591.
9. Cheng Y, Ren J, Wang Z, Mei S, Zhou J (2012) Attributes Union in CP-ABE Algorithm for Large Universe Cryptographic Access Control. 180–186 p.
10. Li J, Wang H, Zhang Y, Shen J (2016) Ciphertext-Policy Attribute-Based Encryption with Hidden Access Policy and Testing. *Ksii Transactions on Internet & Information Systems* 10: 3339–3352.
11. Odelu V, Das AK, Rao YS, Kumari S, Khan MK, Choo KKR (2016) Pairing-based CP-ABE with constant-size ciphertexts and secret keys for cloud environment. *Computer Standards & Interfaces* 54: 3–9.
12. Shynu PG, Singh KJ (2017) An Enhanced CP-ABE Based Access Control Algorithm for Point to Multi-Point Communication in Cloud Computing. *Journal of Information Science & Engineering* 33.
13. Dan B, Crescenzo GD, Ostrovsky R, Persiano G (2003) Public Key Encryption with Keyword Search: Springer Berlin Heidelberg. 506–522 p.
14. Li J, Wang Q, Wang C, Cao N, Ren K, Lou W. Fuzzy keyword search over encrypted data in cloud computing; 2010. pp. 441–445.
15. Shuang LI, Yuan D (2013) Anonymous identity based public key encryption with keyword search. *Computer Engineering & Design* 49: 506–522.
16. Kamara S, Lauter K. Cryptographic Cloud Storage; 2010. pp. 136–149.
17. Cao N, Wang C, Li M, Ren K, Lou W (2011) Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data. *IEEE INFOCOM*: 829–837.
18. Hu C, He P, Liu P (2012) Public Key Encryption with Multi-keyword Search: Springer Berlin Heidelberg. 568–576 p.
19. Miao Y, Ma J, Liu X, Wei F, Liu Z (2016) m^2 -ABKS: Attribute-Based Multi-Keyword Search over Encrypted Personal Health Records in Multi-Owner Setting. *Journal of Medical Systems* 40: 246. <https://doi.org/10.1007/s10916-016-0617-z> PMID: 27696175
20. Huang H, Jianpeng DU, Dai H, Wang R (2017) Multi-sever Multi-keyword Searchable Encryption Scheme Based on Cloud Storage. *Journal of Electronics & Information Technology*.
21. Yu S, Wang C, Ren K, Lou W. Achieving secure, scalable, and fine-grained data access control in cloud computing; 2010. pp. 534–542.
22. Yang K, Jia X, Ren K. Attribute-based fine-grained access control with efficient revocation in cloud storage systems; 2013. pp. 523–528.
23. Xiong AP, Xu CX, Gan QX. A CP-ABE scheme with system attributes revocation in cloud storage; 2014. pp. 331–335.
24. Chow SSM. A Framework of Multi-Authority Attribute-Based Encryption with Outsourcing and Revocation; 2016. pp. 215–226.
25. Liu H, Zhu P, Chen Z, Zhang P, Jiang ZL. Attribute-Based Encryption Scheme Supporting Decryption Outsourcing and Attribute Revocation in Cloud Storage; 2017. pp. 556–561.
26. Wang S, Ye J, Zhang Y (2018) A keyword searchable attribute-based encryption scheme with attribute update for cloud storage. *PLOS ONE* 13: e0197318. <https://doi.org/10.1371/journal.pone.0197318> PMID: 29795577
27. Guo W, Dong X, Cao Z, Shen J (2017) Efficient Attribute-Based Searchable Encryption on the Cloud Storage.

28. Guo F, Mu Y, Susilo W, Wong DS, Varadharajan V (2014) CP-ABE With Constant-Size Keys for Light-weight Devices. *Information Forensics & Security IEEE Transactions on* 9: 763–771.
29. Dan B, Boyen X, Shacham H (2004) *Short Group Signatures*: Springer Berlin Heidelberg. 41–55 p.
30. Shparlinski I (2011) *Computational Diffie-Hellman Problem*: Springer US. 240–244 p.
31. Li R, Zheng D, Zhang Y, Su H, Yang M, Lang P. *Attribute-Based Encryption with Multi-keyword Search*; 2017. pp. 172–177.
32. Zhong H, Zhu W, Xu Y, Cui J (2016) Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage. *Soft Computing* 22: 1–9.
33. Li J, Lin X, Zhang Y, Han J (2016) KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage. *IEEE Transactions on Services Computing* PP: 1–1.
34. Cui H, Deng RH, Wu G, Lai J (2016) An Efficient and Expressive Ciphertext-Policy Attribute-Based Encryption Scheme with Partially Hidden Access Structures. *Computer Networks* 133: 157–165.
35. Liu JK, Yuen TH, Zhang P, Liang K. *Time-Based Direct Revocable Ciphertext-Policy Attribute-Based Encryption with Short Revocation List*; 2018. pp. 516–534.
36. Duquesne S, Lange T (2005) Pairing-based cryptography. *Mathiscernetin volume* 22: 573–590.