Fine-Grained and Traceable Key Delegation for Ciphertext-Policy Attribute-Based Encryption

Jiajie Du[1] and Nurmamat HelIl[1,*]
[1]College of Mathematics and System Science, Xinjiang University
Urumqi 830046, China
[e-mail: dujer1996721@126.com;nur924@sina.com]
[*]Corresponding author: Nurmamat Helil

Received December 4, 2020; revised May 2, 2021; revised June 6, 2021; accepted July 27, 2021; published September 30, 2021

Abstract

Permission delegation is an important research issue in access control. It allows a user to delegate some of his permissions to others to reduce his workload, or enables others to complete some tasks on his behalf when he is unavailable to do so. As an ideal solution for controlling *read* access on outsourced data objects on the cloud, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) has attracted much attention. Some existing CP-ABE schemes handle the *read* permission delegation through the delegation of the user's private key to others. Still, these schemes lack the further consideration of granularity and traceability of the permission delegation. To this end, this article proposes a flexible and fine-grained CP-ABE key delegation approach that supports white-box traceability. In this approach, the key delegator first examines the relations between the data objects, *read* permission thereof that he intends to delegate, and the attributes associated with the access policies of these data objects. Then he chooses a minimal attribute set from his attributes according to the principle of least privilege. He constructs the delegation key with the minimal attribute set. Thus, we can achieve the shortest delegation key and minimize the time of key delegation under the premise of guaranteeing the delegator's access control requirement. The Key Generation Center (KGC) then embeds the delegatee's identity into the key to trace the route of the delegation key. Our approach prevents the delegatee from combining his existing key with the new delegation key to access unauthorized data objects. Theoretical analysis and test results show that our approach helps the KGC transfer some of its burdensome key generation tasks to regular users (delegators) to accommodate more users.

Keywords: CP-ABE, permission delegation, key delegation, minimal attribute set, white-box traceability.

This research work was supported by National Natural Science Foundation of China (grant numbers: 61862059, 61562085).

1. Introduction

Cloud storage technology is increasingly used in the storage, management, and sharing of data. More and more users and organizations are storing their data in the cloud, and are willing to share those data with others. As users lose direct control over data, security and privacy issues are of great concern. Sahai et al. first proposed the concept of Attribute-Based Encryption (ABE)[1]to handle the security of shared data. ABE enables the data owner to specify more flexible access control policies, and is considered one of the most prominent *read* access control mechanisms in cloud storage systems. ABE has two extensions: Key-Policy ABE (KP-ABE)[2] and ciphertext-policy ABE (CP-ABE)[3]. In CP-ABE, the access structure is embedded in the ciphertext, and the user's key is described using attributes. After a data owner encrypts the data object, the corresponding ciphertext can be decrypted when attributes in a user's attribute set satisfy the access structure of the data object. Therefore, CP-ABE is more fit for controlling the *read* permission in cloud storage systems.

Permission delegation [4-18], by which users shift their permissions to others, is an important research topic in access control systems. Some or all permissions can be delegated to other users, thus allowing others to be granted these permissions to perform specific tasks on behalf of the authorized user. CP-ABE is an appropriate solution for making *read* access control on outsourced data objects on the cloud. It handles the *read* permission authorization of data objects through the authorization of the user's private key. Therefore, the delegation of permissions in CP-ABE is equivalent to the delegation of the private keys of the users. In other words, the CP-ABE scheme delegates *read* permissions to data objects through key delegation.

Key delegation enables users to engage in resource sharing and collaboration to complete tasks, thereby improving work efficiency. For example, if a user (delegator) is unable to complete a task due to a business trip or heavy workload, he can delegate his private key to another user so that the other user can complete some tasks on the delegator’s behalf. The CP-ABE [3] has a key delegation mechanism, but it only introduces how to use a subset of a user's attribute set to construct a new key for delegation. There is no detailed explanation of how to choose the attribute set to build the key, or how to delegate the key and to whom. It also lacks further consideration of the granularity and traceability of key delegation.

Generally, traceable CP-ABE schemes [19-25] have private key traceable features. In the practical application of CP-ABE, it is difficult to trace the key owner when a malicious authorized user reveals his or her key; tracing the owner of the delegation key is also difficult. Therefore, it is reasonable to embed the key (original key or delegation key) owner’s identity into the key, and add its traceability to the CP-ABE key delegation mechanism [22-25].

Because the user's private key is associated with his attributes in CP-ABE, this study proposes a flexible and fine-grained key delegation approach for the CP-ABE system, and one that supports white-box traceability. In this approach, the delegator and the key generation center (KGC) construct the delegation key together. The user is responsible for selecting the attribute set that constructs the key; the KGC embeds the identity of the delegatee into the delegation key and calculates the corresponding key components. The key delegation results in the delegation of the *read* permissions to some data objects. Since the KGC participates in constructing a new key by embedding the user's identity into the key, it can supervise the key delegation process and trace the identity of the user who is suspected of leaking keys. In this scheme, a user analyzes the attributes related to the access structures of the data objects to be delegated. According to the principle of least privilege, the user selects the minimal attribute set for construction of the delegation key and delegates the new key to other users. The advantage of this approach is that the user can act as an authorized institution; hence, the entire system can accommodate more users, which helps reduce the workload of KGC. The KGC only generates keys for senior users, whereas the data owner and KGC jointly construct the delegation key. Because the KGC does not need to generate the delegation key from scratch, its workload can be significantly reduced. Our key delegation includes the feature of anti-collusion, which prevents a user from combining his existing keys (original key or delegation keys) to access unauthorized data. It also prevents a user from colluding his keys (original key or delegation keys) with those of other users to access unauthorized data.

The remainder of this article is organized as follows. Section 2 describes related works. Section 3 describes background knowledge related to key delegation, and the components of the key delegation model for CP-ABE. In Section 4, we discuss how to select the minimal attribute set of the key. We introduce the key delegation scheme in Section 5. Section 6 compares different approaches and uses a test to verify the feasibility of the key delegation scheme. Section 7 offers our conclusions, and outlines future research directions.

2. Related work

Many studies on permission delegation in the area of access control research have been conducted [4-18]. [4-14] proposed permission delegation by delegating roles. A role-based delegation model was proposed in [4], which supports hierarchical roles and multistep delegation. [5-8] considered permission delegation of Role-Based Access Control (RBAC) from different perspectives. [9-13] extended RBAC and proposed flexible authorization delegation approaches via role delegation. In [14], a delegation model was proposed to control delegation depth and role scope by establishing a delegation tree. Meanwhile, [15] focused on the permission delegation of identity-based access control, while [16, 17] investigated permission delegation based on attribute-based access control. In the new ID-based access control model [15], authority delegation was realized by creating a delegation token on the delegatee's identity. The relation between delegation and revocation was discussed in detail in the attribute delegation model [16, 17]. [18] presented a non-monotonic multistep permission delegation protocol about workflow and introduced the general characteristics of authorization delegation. Our work is inspired by the permission delegation in access control. We proposed a key delegation approach of CP-ABE in which the *read* permission of data objects can be delegated through key delegation.

The CP-ABE scheme [3] was proposed by Bethencourt et al. As a seemingly ideal solution for controlling *read* access on the outsourced data, it has attracted much attention. In CP-ABE, the access structure is embedded in the ciphertext, and the user's key is described using attributes. A ciphertext can be decrypted when the attributes in the user's key satisfy the access structure of the ciphertext. In addition, a key delegation mechanism is supported in this scheme. However, although a set of attributes is chosen to construct the delegation key, the granularity of the key delegation is not considered. In our proposed method, our main goal is to delegate *read* permission of data in CP-ABE access control. We establish a fine-grained key delegation relying on attribute selection to meet the specific permission delegation needs of the delegator. A user analyzes the attributes related to the access structures of given ciphertext; then, adhering to the principle of least privilege, the user selects a minimal set of attributes that satisfy the access structures of these data objects to construct a new key for delegation.

[19-25] proposed CP-ABE schemes with the feature of traceability. However, these existing traceable CP-ABE schemes have less expressive access control policies; furthermore, the works in [19-25] are limited to "*AND*" gates. Ning et al. [20, 21] proposed a white-box

traceable CP-ABE scheme based on flexible properties. The scheme in [22] has white-box traceability and supports the Linear Secret Sharing Scheme (LSSS) access structure. It is more intuitive than the attribute access structure used in this article. [23-25] proposed CP-ABE schemes to prevent users from abusing delegated keys, but [24, 25] are limited to supporting "*AND*" gates. Our paper supports arbitrary thresholds and achieves similar traceability with [20, 21] via the same Shamir threshold scheme. The cost of traceable storage does not grow linearly with the growth of the number of users. Moreover, [19-25] do not support key delegation, and users must apply for their unique keys directly from the KGC. Key traceability has more practical applications [26-31] in specific environments.

Guan Z et al. proposed a traceable CP-ABE system and enforced key delegation with traceability [32]. However, fine-grained key delegation is not supported in their work. Our CP-ABE key delegation scheme has fine granularity. The basic unit of delegation in this model is ciphertext. The delegatee can only decrypt the delegator's part of ciphertexts with the corresponding delegation key. The delegator aims to control which ciphertexts, within his decryption ability, are to be delegated while under some restrictions. Controlling the delegatee's decryption ability through the delegation key provides flexibility and fine granularity to access control.

[33] proposed an attribute-based proxy re-encryption scheme. It considered re-encrypting a ciphertext with different access policies and assigning the new keys associated with these new ciphertexts to different users. However, the solution in [33] does not support key traceability. Compared to [33], in our work, the KGC embeds the user identity into the key component so that the KGC can trace the identity of the user who possibly leaked the key. In addition, our key delegation scheme is monotonic. The decryption ability of the delegatee with the delegation key will not exceed that of the delegator. We note here that monotonicity of decryption ability is not guaranteed in [33].

[34-35] proposed a CP-ABE scheme with anti-collusion black-box traceability. Our work adds white-box traceability to the CP-ABE key delegation and supports any monotonic access structure. The KGC and the delegator jointly construct the delegation key. Therefore, in our solution, the KGC can not only trace the route of the delegation key but also monitor the key delegation process.

3. Background

3.1 Preliminaries

Access structure: Let set $\{P_1, P_2, ..., P_n\}$ represent a set of entities. For $\forall B, C$, if $B \in \boldsymbol{A}$ and $B \subseteq C$, there is $C \in \boldsymbol{A}$; then, the set $\boldsymbol{A} \subseteq 2^{\{P_1, P_2, ..., P_n\}}$ is monotonic. $\boldsymbol{A} \subseteq 2^{\{P_1, P_2, ..., P_n\}} \setminus \{\varnothing\}$, a non-empty monotonic subset of $\{P_1, P_2, ..., P_n\}$, is called a monotonic access structure.

In this article, entities represent attributes, and thus our access structure is the set of authorized attribute sets. There are two common access structures in CP-ABE: the tree access structure, and LSSS. We choose a tree access structure [3]. We use T to denote the access tree. Root nodes and internal nodes support logical operations such as *OR*, *AND*, and *k of n* $(n > k)$.

Definition 1(Minimal attribute set): Given a tree access structure T of ciphertext, if an attribute set A satisfies the following conditions, then A is called the minimal attribute set of the tree access structure T :

(1) $A \neq \varnothing$;
(2) A satisfies T;
(3) $\forall A' \neq \varnothing$ and $A' \subset A$, T is not satisfied by A'. In other words, A is the smallest set of attributes that meets the access structure T.

We see that there can be multiple minimal attribute sets satisfying a tree access structure T.

Bilinear pair: Let G and G_T be the prime order (p -order) multiplicative cyclic groups with generator g. The mapping e is a bilinear pair if it has the following characteristics:
(1) $\forall u, v \in G$ and $\forall x, y \in Z_p$, $e(u^x, v^y) = e(u, v)^{xy}$;
(2) Non-degeneration: $e(g, g) \neq 1$;
(3) Computability: It is effective to calculate bilinear maps $e: G \times G \rightarrow G_T$.

Decisional Bilinear Diffie–Hellman (DBDH) assumption: Given elements (g, g^a, g^b, g^c, z), where $\forall a, b, c \in Z_p$ and unknown, $z \in G_T$, g is a generator of the cyclic group, determine whether $z = e(g, g)^{abc}$.

Definition 2(Key delegation): A user (delegator) builds a new key and provides it to other users (delegatees); we call this process key delegation. The new key is either created by the delegator's original key or is a key he received from others.

Key delegation allows the delegatee to complete certain tasks on behalf of the delegator with the delegation key. Generally, a delegator performs key delegation when he is unavailable to do these tasks, or if he needs to transfer his tasks to a colleague.

3.2 CP-ABE Key Delegation Scheme and Security Model

3.2.1 CP-ABE Key Delegation Scheme

The traditional CP-ABE scheme [3] has five basic algorithms: ***Setup***, ***Encrypt***, ***KeyGen***, ***Decrypt***, and ***Delegate***. In a basic CP-ABE key delegation scenario, a delegator simply selects certain attributes for constructing a delegation key. In some situations, the route of a delegation key needs to be traced for security reasons. In this proposed method, we mainly consider a flexible, fine-grained CP-ABE scheme with traceability. First, we list the algorithms in this scheme in **Table 1**.

Table 1. Algorithms in CP-ABE key delegation scheme

Algorithm	Input	Output
Setup	security parameter λ	public key PK, master key MK
Encrypt	public key PK, data object m, access structure T	ciphertext CT
KeyGen	public key PK, master key MK, attribute set S, user's identity id	private key SK
Delegate	delegatee's identity id', private key SK, attribute set $S'(S' \subseteq S)$	private key SK'

Decrypt	public key PK, ciphertext CT, private key SK	data object m
Trace	public key PK, private key SK	user's identity id

3.2.2 Security Model and Tracing Model

Security Model: We define a security model through interactive security games between an adversary and challenger. The game is an indistinguishability chosen plaintext attack (IND-CPA) game under the selective access policy. CPA are powerful types of attack, and if an algorithm can resist this type of attack, it can also resist other types of attack.

Our security model is similar to that in [25], but instead of only supporting the "*AND*" gate access structure, this model supports an arbitrary threshold.

Definition 3 We call the traceable key delegation scheme selectively secure and the IND-CPA secure if in the security game the adversary has at most a negligible advantage in polynomial time.

Tracing Model: The key tracing model of the CP-ABE key delegation scheme that supports dynamic attribute space traceability is similar to the tracing model of [32]. However, in this article, the threshold scheme is used to record user identity.

Definition 4 We call the solution in this article traceable if the advantage obtained by the adversary in the traceability game in any polynomial time is negligible relative to the security parameter λ.

3.3 CP-ABE Key Delegation Model

We note that in CP-ABE, the delegation of the *read* permission of a data object is achieved through key delegation. We consider the key delegation of the CP-ABE based on permission delegation and provide a formal definition to the key delegation model. We design the key delegation model according to the requirements that follow.

(1) The delegation is monotonic key delegation. The decryption ability of the delegatee with his delegation key, which originated from a key of the delegator, does not exceed that of the delegator with the very same key.
(2) The model supports multistage key delegation. If the delegatee wants to further delegate the key to other users, he can act as a delegator and initiate key delegation with his delegation key.
(3) The model supports flexible and fine-grained delegation. The basic unit of delegation is essentially the ciphertext. The delegatee can only decrypt specific ciphertexts with the delegation key. The delegator aims to control which ciphertexts, within his decryption ability, are to be decrypted by the delegatee, under the premise that some other ciphertexts are not to be decrypted.
(4) The model prevents the delegatee from combining the delegation keys with any keys he owns to access additional data objects.

In addition, we have the following assumptions:

(1) To ensure security, a user cannot directly deliver his key to others as a delegation key without going through the delegation key generation process.
(2) The delegator is unaware of the original decryption ability of the delegatee. The delegator only cares about the decryption ability of the delegatee with the delegation key.

We first establish the key delegation model of CP-ABE as follows:

(1) U, A, K, and $\boldsymbol{CT}$ represent users, attribute sets, keys, and ciphertext sets, respectively;

(2) $UA \subseteq U \times 2^A$: a mapping user-to-attribute set assignment relation;

(3) $UAK \subseteq UA \times K$: a mapping of users (associated with attribute sets) to private keys;

(4) $KH \subseteq K \times K$ is a partial order on K named the key hierarchy relation and is recorded as $\geq$. For two given keys $SK, SK' \in K$, SK is constructed by attribute set S, and SK' constructed by attribute set S'; $SK \geq SK'$ only if $S' \subseteq S$ for $S, S' \subseteq A$, and SK' is generated from SK. Ciphertexts that can be decrypted by SK' can also be decrypted by SK;

(5) $DK = \{\langle (u_0, S, SK), (u_1, S', SK') \rangle \mid SK \geq SK', (u_0, S, SK), (u_1, S', SK') \in UAK\}$ represents a mapping key-to-key delegation relation.

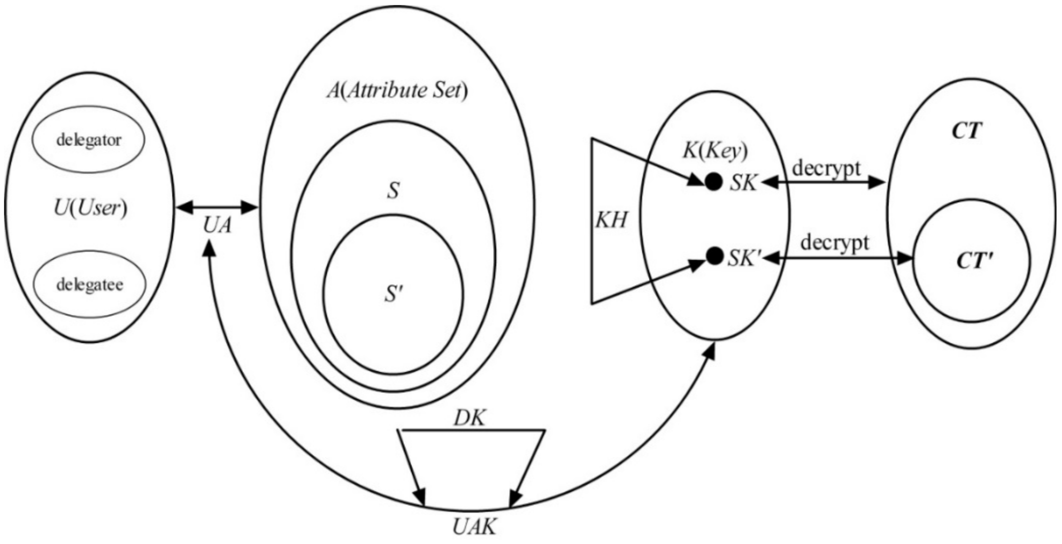

Fig. 1. Key delegation model for CP-ABE

Fig. 1 illustrates the components of the CP-ABE key delegation model. Suppose a user (delegator) u_0 possesses attributes S, that is, $(u_0, S) \in UA$. The KGC generates an original key SK for the delegator u_0 according to his attribute set S; then, there is a key assignment relationship $(u_0, S, SK) \in UAK$. u_0 can decrypt all the ciphertexts in $\boldsymbol{CT} = \{CT_1, CT_2, ..., CT_n\}$ with SK since his attributes match access structures of ciphertexts in $\boldsymbol{CT}$. u_0 selects some attributes S' ($S' \subseteq S$) to construct the key SK' according to the access structures of the ciphertext in $\boldsymbol{CT'}(\boldsymbol{CT'} \subseteq \boldsymbol{CT})$, and delegates SK' to the user (delegatee) u_1. Then, we have the key assignment relation $(u_1, S', SK') \in UAK$.

Consequently, the key delegation relation $\langle (u_0, S, SK), (u_1, S', SK') \rangle \in DK$ is established. Since S is associated with SK and S' is associated with SK' and $S' \subseteq S$, there is a key hierarchy relation $(SK, SK') \in KH$ between SK and SK'. The ciphertext in $\boldsymbol{CT'}(\boldsymbol{CT'} \subseteq \boldsymbol{CT})$ can be decrypted with SK'.

This key delegation model supports multistage delegation. User u_1 receiving delegation key SK' can still be a delegator to forward the delegation key SK' to other users with certain alternations. When a user acts as a delegator, he can delegate a key to multiple users.

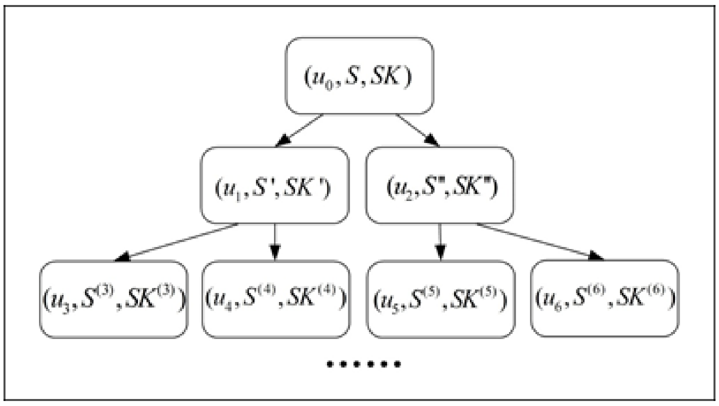
(u_0, S, SK)
(u_1, S', SK')
(u_2, S'', SK'')
$(u_3, S^{(3)}, SK^{(3)})$
$(u_4, S^{(4)}, SK^{(4)})$
$(u_5, S^{(5)}, SK^{(5)})$
$(u_6, S^{(6)}, SK^{(6)})$
......

Fig. 2. Example of key delegation paths

The key delegation path starts from a primary user, and after multiple delegation steps, it can form a delegation tree as in **Fig. 2**. Multiple delegation paths are reflected in the delegation tree, such as

$$(u_0, S, SK) \to (u_1, S', SK') \to (u_3, S^{(3)}, SK^{(3)}) \to \ldots$$
$$(u_0, S, SK) \to (u_1, S', SK') \to (u_4, S^{(4)}, SK^{(4)}) \to \ldots$$
$$(u_0, S, SK) \to (u_2, S'', SK'') \to (u_5, S^{(5)}, SK^{(5)}) \to \ldots$$
$$(u_0, S, SK) \to (u_2, S'', SK'') \to (u_6, S^{(6)}, SK^{(6)}) \to \ldots$$

In the key delegation tree, the attribute set on each node is a subset of the attribute set of its parent node, e.g., $S' \subseteq S$, $S'' \subseteq S$, and $S^{(3)} \subseteq S'$. There are also key hierarchy relations between the key of a parent node and the key of its child nodes, e.g., $SK \geq SK'$, $SK' \geq SK^{(3)}$, and $SK' \geq SK^{(4)}$. The ciphertexts with the key on the child node can also be decrypted by the key on its parent node. The delegation tree also reflects the fact that the key delegation supports monotonic multistage delegation. That is to say, the decryption ability of the delegatee with the delegation key SK' does not exceed the decryption ability of the delegator with the original key SK.

4. Attribute extraction for key delegation

In a CP-ABE key delegation scenario, a key delegator wants a delegatee to decrypt certain ciphertexts using the delegation key. In this section, we present how the delegator selects the minimal set of attributes that satisfies ciphertext access structures in a given ciphertext set by following the principle of least privilege.

4.1 Minimal Attribute Set Family of a Ciphertext

We first consider minimal attribute sets for a ciphertext. Here, we design an algorithm to generate a minimal attribute set family that meets the tree access structure of a ciphertext. The algorithm is shown in **Table 2**.

Table 2. Algorithm for selecting the minimal attribute set family

```
Input: Tree access structure: $T$. ( $R$ is the root node of $T$. $Q$ are non-root nodes of $T$ ).
Output: Minimal attribute set family $S$.
Begin:
1   $S=\varnothing$;
2   IF $R=AND(Q_1,Q_2,...,Q_n)$   THEN
3       $S=\{\{Q_1,Q_2,...,Q_n\}\}$
4   ELSE IF $R=OR(Q_1,Q_2,...,Q_n)$   THEN
5       $S=\{\{Q_1\},\{Q_2\},...,\{Q_n\}\}$
6   ELSE IF $R=k\ of\ n(Q_1,Q_2,...,Q_n)$   THEN
7    $S=\{\{Q_{1_1},Q_{1_2},...,Q_{1_k}\},\{Q_{2_1},Q_{2_2},...,Q_{2_k}\},...,\{Q_{c_1},Q_{c_2},...,Q_{c_k}\}\}(c=C_n^k)$
8    ELSE RETURN $S$
9   FOR All $S_i$ IN $S$ DO
10          FOR All $Q$ IN $S_i$ DO
11              IF $Q=AND(Q_1,Q_2,...,Q_n)$   THEN
12                      $temp=(S_i\setminus\{Q_i\})\cup\{Q_1,Q_2,...,Q_n\}$
13                      $S=(S\setminus\{S_i\})\cup\{temp\}$
14              ELSE IF $Q=OR(Q_1,Q_2,...,Q_n)$   THEN
15                      $temp=S_i\setminus\{Q\}$
16                      $S=S\setminus\{S_i\}$
17                      FOR $i=1$ to $n$
18                              $temp_i=temp\cup\{Q_i\}$
19                              $S=S\cup\{temp_i\}$
20              ELSE IF $Q=k\ of\ n(Q_1,Q_2,...,Q_n)$   THEN
21                              $temp=S_i\setminus\{Q\}$
22                              $S=S\setminus\{S_i\}$
23                              FOR $i=1$ to $C_n^k$
24                                      $temp_i=temp\cup\{Q_{i_1},Q_{i_2},...,Q_{i_k}\}$
25                                      $S=S\cup\{temp_i\}$
26          ELSE RETURN $S$
27  RETURN $S$
End
```

Table 2 shows the algorithm for calculating the minimal attribute set family according to the access structure. In this algorithm, lines 2–8 handle the root node; different nodes receive a different minimal attribute set family. Lines 9–27 traverse the internal nodes in the set family. If the node is "*AND*", lines 11–13 will be executed. If the node is "*OR*", lines 14–19 will be executed. If the node is "*k of n*", lines 20–25 will be executed. All of the nodes are traversed until all of the sets are only composed of attributes, and then the minimal attribute set family will be constructed.

We further illustrate this algorithm with the example in Fig. 3.

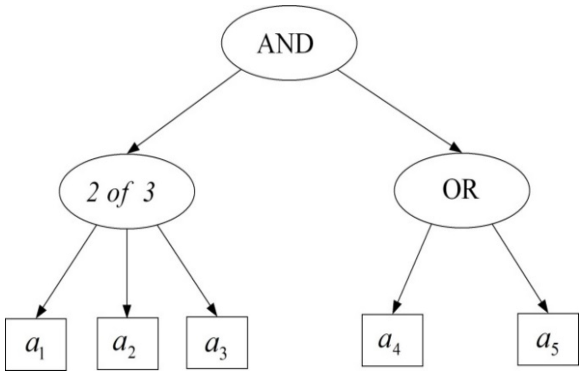

Fig. 3. Example of tree access structures

The algorithm calculates the minimal attribute set family $S=\{\{a_1,a_2,a_4\},\{a_1,a_2,a_5\},\{a_2,a_3,a_4\},\{a_2,a_3,a_5\},\{a_1,a_3,a_4\},\{a_1,a_3,a_5\}\}$ for the access structure shown in Fig. 3.

4.2 Attribute selection for delegation key establishment

Based on minimal attribute sets of a ciphertext, we now discuss how to determine a set of attributes that is most suitable for the delegation key establishment.

Suppose delegator *Alice*'s private key SK is constructed with S, and S satisfies access structures of all of the ciphertexts in $\boldsymbol{CT}=\{CT_1,CT_2,...,CT_n\}$. That is, *Alice* can decrypt all of the ciphertexts in $\boldsymbol{CT}$ with SK.

Now *Alice* wants delegatee *Bob* to decrypt ciphertexts in $\boldsymbol{CT'}=\{CT_1,CT_2,...,CT_k\}$ by using the delegation key to complete some job tasks on her behalf. This is *Alice*'s primary intention. However, *Alice* may also have additional restrictions on the delegation key. For example, due to a security concern, *Alice* may not want *Bob* to decrypt any ciphertext in $\boldsymbol{CT''}$, $\boldsymbol{CT''}\subseteq\boldsymbol{CT}\setminus\boldsymbol{CT'}$, or she may follow the least privilege principle. That is, a few of the remaining ciphertexts in $\boldsymbol{CT}_r=\boldsymbol{CT}\setminus\boldsymbol{CT'}$ can be decrypted as possible by the decryption key. Therefore, according to her specific access control requirement, *Alice* tries to construct a key SK' for delegation.

Since $\boldsymbol{CT'}$ is *Alice*'s primary concern, she needs to determine a set of attributes that satisfies all access structures of ciphertexts in $\boldsymbol{CT'}$. *Alice* connects the access structures of all of the ciphertexts in $\boldsymbol{CT'}$ with an "*AND*" gate to obtain a new access tree $\boldsymbol{T'}$. We denote the access structure corresponding to each ciphertext $CT_i\in\boldsymbol{CT'}$, $i=1,2,...,k$ as T_i, $i=1,2,...,k$. The new access tree $\boldsymbol{T'}$ obtained is shown in Fig. 4.

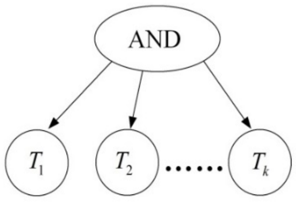

Fig. 4. New access structure $\boldsymbol{T'}$

Alice calculates the minimal attribute sets for $\boldsymbol{T'}$ using the algorithm in **Table 2**. We denote minimal attribute sets as $\mathbf{A}_1', \mathbf{A}_2', ..., \mathbf{A}_{n_k}'$. From the construction of $\boldsymbol{T'}$, we can see that each $\mathbf{A}_j', j \in \{1, 2, ..., n_k\}$ meets the access structures of all of the ciphertexts in $\boldsymbol{CT'}$. Here, we discuss different cases, taking into account *Alice*'s additional concerns.

Case 1: Decryption of all of the ciphertexts in CT' must be guaranteed.

Bob must have the privilege to decrypt all of the ciphertexts in $\boldsymbol{CT'}$. *Alice* does not care if *Bob* can decrypt the remaining ciphertexts in $\boldsymbol{CT}$. *Alice* chooses the smallest set from $\mathbf{A}_1', \mathbf{A}_2', ..., \mathbf{A}_{n_k}'$, and assigns it to S'. She uses S' to construct a new key.

The key constructed by S' ensures that *Bob* can decrypt all of the ciphertexts in $\boldsymbol{CT'}$. Because *Alice* selects the smallest set from $\mathbf{A}_1', \mathbf{A}_2', ..., \mathbf{A}_{n_k}'$ to assign to S', she achieves the smallest size delegation key.

Case 2: Decryption of all of the ciphertexts in CT' must be guaranteed; the number of ciphertexts from the remaining set $CT_r = CT \setminus CT'$ can be decrypted as little as possible.

On the basis of Case 1, *Alice* further checks which $\mathbf{A}_j', j \in \{1, 2, ..., n_k\}$ is most suitable for key construction. For each $\mathbf{A}_j', j \in \{1, 2, ..., n_k\}$, she examines ciphertexts in $\boldsymbol{CT}_r$ access structures thereof that can be satisfied by $\mathbf{A}_j'$. Let $l_j, j \in \{1, 2, ..., n_k\}$ be the number of ciphertexts in $\boldsymbol{CT}_r$, and access structures thereof can be met by $\mathbf{A}_j'$. *Alice* sets $S' = \mathbf{A}_*'$, where $l_* = \min(l_1, l_2, ..., l_{n_k})$. If $\exists l_j = l_*, |\mathbf{A}_j'| \neq |\mathbf{A}_*'|$, then *Alice* chooses a smaller set from $\mathbf{A}_*'$ and $\mathbf{A}_j'$, and assigns it to S'. *Alice* then uses S' to construct a new key.

Using S' to construct a key can not only guarantee that *Bob* is able to decrypt all of the ciphertexts in $\boldsymbol{CT'}$, but that he can only decrypt the least amount of ciphertexts in $\boldsymbol{CT}_r$. *Alice* also obtains the smallest size delegation key.

Case 3: Decryption of all ciphertexts in CT' must be guaranteed; must not decrypt any ciphertext in CT'', $CT'' \subseteq CT_r$.

On the basis of Case 1, *Alice* further checks if there exists $\mathbf{A}_j', j \in \{1, 2, ..., n_k\}$ that is suitable for key construction. For each $\mathbf{A}_j', j \in \{1, 2, ..., n_k\}$, she examines ciphertexts in $\boldsymbol{CT''}$ access structures thereof that can be satisfied by $\mathbf{A}_j'$. Let $l_j, j \in \{1, 2, ..., n_k\}$ be the number of ciphertexts in $\boldsymbol{CT''}$, and access structures thereof can be met by $\mathbf{A}_j'$. *Alice* sets $S' = \mathbf{A}_*'$, where $l_* = 0$. If $\exists l_j = l_* = 0, |\mathbf{A}_j'| \neq |\mathbf{A}_*'|$, then *Alice* chooses a smaller set from $\mathbf{A}_*'$ and $\mathbf{A}_j'$, and assigns it to S'. *Alice* uses S' to construct a new key. If $l_* = \min(l_1, l_2, ..., l_{n_k}) \neq 0$, then key delegation cannot be performed.

In this way, the key constructed with S', if possible, can not only guarantee that *Bob* is able to decrypt all of the ciphertexts in $\boldsymbol{CT'}$ but also that he cannot decrypt the ciphertexts in $\boldsymbol{CT''}$. In addition, *Alice* obtains the smallest size delegation key.

As an example, assume that *Alice* can decrypt ciphertexts in $\boldsymbol{CT} = \{CT_1, CT_2, CT_3, CT_4\}$ with her private key. Access structures of ciphertexts CT_1, CT_2, CT_3, and CT_4 are as follows:

$$T_1 = a_1 \ AND \ (a_2 \ OR \ a_3),$$

$$T_2 = (a_4 \ AND \ a_5) \ OR \ a_6,$$

$$T_3 = a_1 \ AND \ (a_3 \ OR \ a_5),$$

$$T_4 = a_4 \ AND \ a_5.$$

As her primary concern, *Alice* wants *Bob* to decrypt ciphertexts in $\boldsymbol{CT'} = \{CT_1, CT_2\}$ through key delegation.

Alice uses the "*AND*" gate to connect T_1 and T_2 to get $\boldsymbol{T'}$; then, she inputs $\boldsymbol{T'}$, and executes the algorithm in **Table 2** to find the minimal attribute sets. The minimal attribute sets are $\mathbf{A}_1' = \{a_1, a_2, a_4, a_5\}$, $\mathbf{A}_2' = \{a_1, a_3, a_4, a_5\}$, $\mathbf{A}_3' = \{a_1, a_2, a_6\}$, and $\mathbf{A}_4' = \{a_1, a_3, a_6\}$.

Case 1. Since $|\mathbf{A}_1'| = |\mathbf{A}_2'| > |\mathbf{A}_3'| = |\mathbf{A}_4'|$, *Alice* chooses $\mathbf{A}_3'$ or $\mathbf{A}_4'$, assigns it to S', and uses S' to construct the key. The delegation key ensures that *Bob* is able to decrypt ciphertexts in $\boldsymbol{CT'}$.

Case 2. According to the minimal attribute sets, *Alice* uses these four attribute sets to match access structures of CT_3 and CT_4, and obtains $l_1 = 2$, $l_2 = 2$, $l_3 = 0$, and $l_4 = 1$. Then, *Alice* obtains $l_* = \min(l_1, l_2, l_3, l_4) = l_3 = 0$. $\mathbf{A}_3'$ meets the requirements of Case 2. Then, we set $S' = \mathbf{A}_3'$. *Alice* uses S' to construct the key.

Case 3. *Alice* has an additional requirement: she does not want *Bob* to decrypt $\boldsymbol{CT''} = \{CT_3, CT_4\}$. According to Case 2, $l_1 = 2$, $l_2 = 2$, $l_3 = 0$, $l_4 = 1$, $l_* = \min(l_1, l_2, l_3, l_4) = l_3 = 0$, and $|\mathbf{A}_1'| = |\mathbf{A}_2'| > |\mathbf{A}_3'| = |\mathbf{A}_4'|$. Therefore, $\mathbf{A}_3'$ meets the requirements of Case 3. Finally, *Alice* chooses $\mathbf{A}_3'$, and assigns it to S' to construct the key.

5. CP-ABE Key delegation scheme

5.1 Scheme Construction

In this section, we describe the fine-grained, traceable CP-ABE key delegation scheme. The KGC and the delegator construct the delegation key. We go through a key delegation process between two users who play the roles of the delegator and the delegatee, respectively. The framework of our scheme, including the key delegation process, is shown in **Fig. 5**.

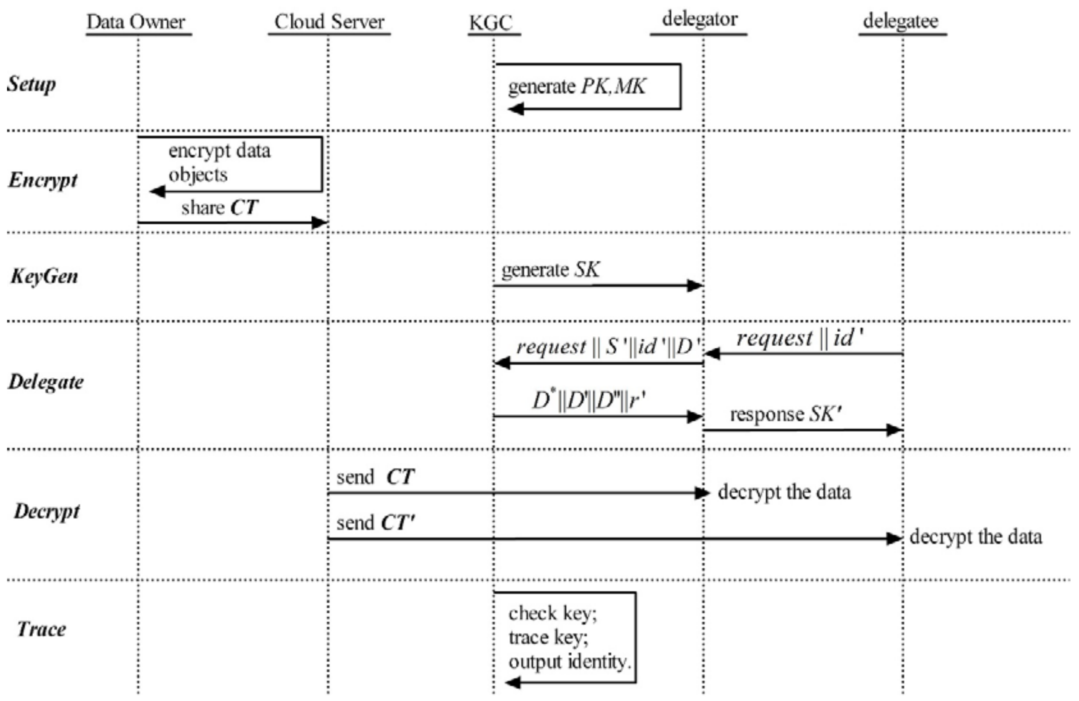

Fig. 5. Framework of our scheme

Fig. 5 depicts the proposed scheme, which consists of six algorithms. The KGC performs ***Setup*** to generate PK and MK for the whole system. Then, according to the attribute set S and user id of the user (delegator), ***KeyGen*** generates key SK for the delegator; meanwhile, the data owner executes ***Encrypt*** to encrypt data objects and shares the ciphertexts on the cloud server. The delegatee sends a request with his identity $request \| id'$ for the delegated key. The delegator and KGC jointly execute ***Delegate*** . The KGC receives $request \| S' \| id' \| D'$ from the delegator, embeds the identity of the delegatee into the delegation key, calculates the corresponding key component, and returns it to the delegator. The delegator uses the selected minimal attribute set and $D^* \| D'' \| r'$ provided by the KGC to create the delegation key SK', and sends it to the delegatee. This delegation allows the delegatee to use ***Decrypt*** to decrypt the ciphertexts in the $\boldsymbol{CT'}$ with the delegated key SK'. The delegator decrypts ciphertexts in $\boldsymbol{CT}$ with his original key SK. If a malicious user leaks the key, the KGC executes ***Trace*** to trace the identity of the user corresponding to the leaked key. The delegator examines the attributes related to the access structures of the ciphertexts, and selects a minimal attribute set using the method presented in Section 4.2. The details of the delegation are as follows.

Setup**:** The algorithm $\boldsymbol{Setup}(\lambda) \to (PK, MK)$ is executed by the KGC. First, it inputs security parameters λ to algorithm $(p, g, G, G_T, e) \leftarrow \boldsymbol{g}(\lambda)$ to generate groups, where G and G_T are the p-order groups with a generator g, and $e: G \times G \to G_T$ is the bilinear map. We use a hash function $H: \{0,1\}^* \to G$ and model it as a random oracle. The KGC randomly selects $\alpha, \beta \in Z_p$ and calculates $e(g,g)^\alpha$. We use a symmetric key encryption algorithm. Let two different symmetric keys be k_1 and k_2. The public key is $PK=(G,g,g^{\beta},e(g,g)^{\alpha})$, and the master key is $MK=(g^{\alpha},\beta,k_1,k_2)$. The KGC uses a $t-1$ degree polynomial $v=f(\mu)$ to initialize the Shamir (t,n) threshold scheme. When the key generation algorithm is executed, the KGC processes the user identity information and achieves points $\{(\mu_i, f(\mu_i))\}$ on the polynomial. How to achieve the points will be described in more detail in the key generation algorithm.

Encrypt **:** The data owner executes the encryption algorithm $\boldsymbol{Encrypt}(PK,T,m)\rightarrow CT$. He inputs the system public key PK, a data object $m\in G_T$, and an access tree T. The algorithm first chooses a polynomial q_x for each node x in the tree T. The algorithm randomly selects $s\in Z_p$ starting with the root node R and makes $q_R(0)=s$. Y represents the set of leaf nodes for $\forall y\in Y$, and $att(y)$ represents the attribute of node y. The algorithm outputs the following ciphertext:

$$CT=(T,\tilde{C}=me(g,g)^{\alpha s},C_0=g^{s},C_0{}'=g^{\beta s},$$
$$\forall y\in Y: C_y=g^{q_y(0)},C_y{}'=H(att(y))^{q_y(0)}). \quad (1)$$

KeyGen **:** The KGC executes $\boldsymbol{KeyGen}(PK,MK,id,S)\rightarrow SK$. It inputs PK, MK, an attribute set S, and the user's identity id. The KGC then calculates

$$\mu=Enc_{k_1}(id),\ v=f(\mu),\ c=Enc_{k_2}(\mu\,\|\,v). \quad (2)$$

The distribution of μ is indistinguishable from a random number and f is a polynomial of a linear transformation. Therefore, the distributions of v and c are no different from random numbers. The KGC selects a random number $r\in Z_p$ for the user, and for $\forall a_j\in S$ it randomly selects $r_j\in Z_p$. The KGC then calculates

$$D=g^{(\alpha+r)/(\beta+c)},\ D_j=g^{r}\cdot H(a_j)^{r_j},D_j{}'=g^{r_j}. \quad (3)$$

The key output of the algorithm is

$$SK=(D=g^{\frac{\alpha+r}{\beta+c}},D'=c,\forall a_j\in S: D_j=g^{r}\cdot H(a_j)^{r_j},D_j{}'=g^{r_j})\cdot \quad (4)$$

From the perspective of the key delegation model, we have now the assignment relations about the delegator, such as

$$(u_0,S)\in UA,\ (u_0,S,SK)\in UAK. \quad (5)$$

Delegate **:** The algorithm $Delegate(SK,S',id')\rightarrow SK'$ is jointly completed by the KGC and the delegator. A user (delegatee) u_1 sends a request with his identity $request\,\|\,id'$ to the delegator u_0. After u_0 receives it, he tries to send the delegatee a key that can only decrypt the ciphertexts in the ciphertext set $\boldsymbol{CT'}(\boldsymbol{CT'}\subseteq \boldsymbol{CT})$. Therefore, u_0 examines the access structures of ciphertexts in $\boldsymbol{CT'}$, finds the minimal attribute set that satisfies access structures of ciphertexts in $\boldsymbol{CT'}$, and records it as $S'(S'\subseteq S)$ (further details for determining the minimal attribute set are provided in Section 4.2). u_0 sends $request\,\|\,S'\|id'\|D'$ to the KGC. Based on the delegation request, the KGC calculates

$$\mu'=Enc_{k_1}(id'),\ v'=f(\mu'),\ c'=Enc_{k_2}(\mu'\,\|\,v'). \quad (6)$$

The KGC saves point (μ', v') and selects a random number r' for u_1. The KGC sends $D^* = g^{(\alpha+r+r')/(\beta+c)} \| D' \| D'' = c' \| r'$ to u_0. u_0 selects a random number r_k' for $\forall a_k \in S'$. The form of the delegation key is as follows:

$$SK' = (D^* = g^{\frac{\alpha+r+r'}{\beta+c'}}, D'' = c',$$
$$\forall a_k \in S': D_k^* = D_k \cdot g^{r'} \cdot H(a_k)^{r_k'}, D_k'' = D_k' g^{r_k'}) \quad (7)$$

Finally, the assignment relation $(u_1, S', SK') \in UAK$ is established. The key delegation relation can be expressed as $<(u_0, S, SK), (u_1, S', SK')> \in DK$.

If u_1 wants to further delegate the key SK'' to another user u_2 to allow him to decrypt ciphertexts in $\boldsymbol{CT}^*$ ($\boldsymbol{CT}^* \subseteq \boldsymbol{CT'}$), u_1 acts as a delegator. The KGC helps u_1 to construct the key. Consequently, the delegation relation $<(u_1, S', SK'), (u_2, S'', SK'')> \in DK$ will be established. A delegation path from u_0 to u_2 will be formed as follows:

$$(u_0, S, SK) \to (u_1, S', SK') \to (u_2, S'', SK''). \quad (8)$$

The key delegation process initiated by the delegator forms multiple delegation paths after multistage and multiple delegations.

***Decrypt*:** The user executes algorithm $Decrypt(PK, SK, CT) \to m$. If the attribute set related to SK matches the access structure T of CT, the algorithm executes the decryption process. If the node x is a leaf node, then let $i = att(x)$, and the recursive algorithm is as follows:

If $i \in S$, then

$$DecryptNode(CT, SK, x) = \frac{e(D_i, C_x)}{e(D_i', C_x')},$$
$$= \frac{e(g^r \cdot H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})},$$
$$= e(g,g)^{rq_x(0)}. \quad (9)$$

If $i \notin S$, then $DecryptNode(CT, SK, x) = \perp$.

Now consider the recursive process of the internal node x. The execution of the algorithm $DecryptNode(CT, SK, x)$ is as follows: for all nodes z that are children of x, $DecryptNode(CT, SK, x)$ is run, and the algorithm outputs F_z.

Let S_x be a k_x-sized set of child nodes z, and let $F_z \neq \perp$. Then, we compute:

$$F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S_x'}(0)},$$
$$= \prod_{z \in S_x} (e(g,g)^{r \cdot q_z(0)})^{\Delta_{i,S_x'}(0)},$$
$$= \prod_{z \in S_x} (e(g,g)^{r \cdot q_{parent(z)}(index(z))})^{\Delta_{i,S_x'}(0)},$$
$$= \prod_{z \in S_x} (e(g,g)^{r \cdot q_x(i)})^{\Delta_{i,S_x'}(0)},$$
$$= e(g,g)^{rq_x(0)}. \quad (10)$$

where $i = index(z)$, and $S_x^{'} = \{index(z) : z \in S_x\}$. Then returns the result above.

If S satisfies the access tree, we make

$$A = DecryptNode(CT, SK, R) = e(g,g)^{rq_R(0)} = e(g,g)^{rs}, \tag{11}$$

$$\begin{aligned} E &= e(D, (C_0)^{D'} C_0'), \\ &= e(g^{\frac{\alpha+r}{\beta+c}}, (g^s)^c g^{\beta s}), \\ &= e(g,g)^{\alpha s} e(g,g)^{rs}, \end{aligned} \tag{12}$$

$$F = \frac{E}{A} = e(g,g)^{\alpha s}, \tag{13}$$

$$m = \frac{\tilde{C}}{F}. \tag{14}$$

The process of decrypting the ciphertext with the delegation key by the delegatee is similar.

Trace **:** If the KGC suspects that a user maliciously leaked the key, then the KGC executes algorithm $\boldsymbol{Trace}(PK, SK) \to id$ and traces the identity of the malicious user. Key tracing involves two steps, as discussed below.

Key integrity check: The KGC first checks the integrity of the key.

First, check if $D' \in Z_p$ and $D \in G$, and for $\forall a_j \in S$, $D_j, D_j' \in G$.

Second, check if

$$\frac{e(D, g^{D'} g^{\beta})}{e(D_j, g) / e(D_j', H(j))} = e(g,g)^{\alpha}. \tag{15}$$

If the key satisfies these two conditions, it is considered to be complete. Otherwise, it is incomplete, and it is unnecessary to trace the incomplete key.

Determination of the ID: If the key is complete, then the KGC executes the decryption algorithm $Dec_{k_2}(D')$ to obtain $\mu^* = \mu$, $\nu^* = \nu$. If $(\mu^*, \nu^*) \in \{(\mu_1, \nu_1), (\mu_2, \nu_2), \ldots, (\mu_{t-1}, \nu_{t-1})\}$, then KGC calculates $Dec_{k_1}(\mu^*)$ to obtain the identity of the user who might have deliberately leaked his key. Otherwise, the KGC uses the point (μ^*, ν^*) with existing $t-1$ points $(\mu_1, \nu_1), (\mu_2, \nu_2), \ldots, (\mu_{t-1}, \nu_{t-1})$ to recover the secret value d_0^* of the threshold scheme. If $d_0^* = f(0)$, then the KGC calculates $Dec_{k_1}(\mu^*)$ to obtain the identity of the user. Otherwise, the KGC outputs $\perp$.

6. Scheme analysis

6.1 Performance analysis

With the criteria of policy expression, key delegation, traceability, and delegation granularity, **Table 3** compares the presented work with some existing works. The schemes in [3], [32], [33], and our proposed scheme, each have a key delegation feature. Still, only our work supports fine-grained key delegation, which is based on ciphertext-level granularity.

[3] is a pioneer work within the CP-ABE domain, while [25], [33], and [32] are all closely related to this study. [25] adds traceability to [3], while [32] does not have traceability but adds

key delegation to [3]. Furthermore, [33] not only supports key delegation but also tracks delegation keys. Meanwhile, the present paper focuses on the granularity of delegation keys, in addition to having the functions of key delegation and key tracking.

Table 3. Comparison of different approaches

scheme	**policy expression**	**key delegation**	**traceability**	**delegation granularity**
reference [3]	arbitrary threshold/Boolean expression	yes	no	---
reference [25]	"*AND*" operation	no	yes	---
reference [32]	arbitrary threshold/Boolean expression	yes	yes	---
reference [33]	arbitrary threshold/Boolean expression	yes	no	---
this article	arbitrary threshold/Boolean expression	yes	yes	fine-grained

Table 4 compares the size of the public parameter, private key, ciphertext, and delegation key in this scheme with those in [3], [25], [32], and [33]. It also compares the regular key/delegation key generation, and the encryption and decryption costs. We use the notations below.

$\exp$: the number of exponential operations;
e : the number of bilinear operations;
L_0 : bit length of the elements in G ;
L_1 : bit length of the elements in G_T ;
ρ : the number of all attributes in the system;
δ : ciphertext-related attribute number;
σ : user key-related attribute number;
θ : delegation key-related attribute number, $\theta \le \sigma$.

Table 4. Comparison of different approaches

scheme	**reference [3]**	**reference [25]**	**reference [32]**	**reference [33]**	**this paper**
public parameter	$4L_0 + L_1$	$(2\rho + 2)L_0 + L_1$	$5L_0 + L_1$	$(3\rho + 2)L_0 + L_1$	$3L_0 + L_1$
private key	$(2\sigma + 1)L_0$	$(\rho + 2)L_0$	$(2\sigma + 1)L_0$	$(2\rho + 1)L_0$	$(2\sigma + 1)L_0$
ciphertext	$(2\delta + 1)L_0$ $+L_1$	$(2\rho - \delta + 2)L_0$ $+L_1$	$(2\delta + 1)L_0$ $+L_1$	$(2 + \rho)L_0$ $+L_1$	$(2\delta + 2)L_0$ $+L_1$

delegation key	$(2\theta+1)L_0$	---	$(2\theta+1)L_0$	$(2\rho+1)L_0$	$(2\theta+1)L_0$
encryption cost	$(2\delta+1)\exp$ $+\delta e$	$(2\rho-\delta+2)$ $\exp+e$	$(2\delta+1)\exp$ $+\delta e$	$(\rho+2)\exp$ $+\rho e$	$(2\delta+2)\exp$ $+\delta e$
decryption cost	$3\exp+e$	$\exp+(\rho+1)e$	$4\exp+e$	$3\exp+(\rho+1)e$	$3\exp+e$
key generation cost	$(2\sigma+1)\exp$	$(\rho+2)\exp$	$(2\sigma+1)\exp$	$(\rho+1)\exp$	$(2\sigma+1)\exp$
delegation key generation cost	$(2\theta+1)\exp$	---	$(2\theta+1)\exp$	$(2\rho+1)\exp$	$(2\theta+1)\exp$

Compared with [3], both the size of the ciphertext and the encryption cost of our scheme are larger; the main reason for this is that our scheme includes the feature of traceability. Compared with other schemes, the size of the delegation key is smallest in our scheme because we determined the least number of attributes that match the access requirement of the delegator.

6.2 Test results

The key delegation scheme was evaluated by conducting a small-scale test. We developed a prototype system of the scheme. We measured the average time cost for algorithms in this scheme. The test environment was a PC with an Intel® Core™ i7-10510U CPU @ 1.80 GHz 2.30 GHz, with 16 GB RAM, and operating on 64-bit Linux Ubuntu 20.04.2.0. The prototype system was developed in Java and run on Java Development Kit (JDK) 15. It was based on the Java Pairing Based Cryptography (JPBC) library 1.2.0 to implement bilinear operations. We chose 128-bit AES for the symmetric encryption algorithm. We set the degree of the polynomial used in the test to be 20.

The number of attributes associated with the ciphertext, delegation key, and original key was the same for all algorithms, and keys met the ciphertext access structure. From **Fig. 6**, we can see the changing trend in the computation time of each algorithm in the scheme with the increase in the number of attributes. Under the same conditions, we run each algorithm 10 times, and the average time of the 10 runs was taken as our final test result.

It can be seen from **Fig. 6** that the time cost of our scheme for ***Setup*** and ***Delegate*** (the KGC part) does not increase with the increase in the number of attributes; this is because the two algorithms are not relevant to attributes.

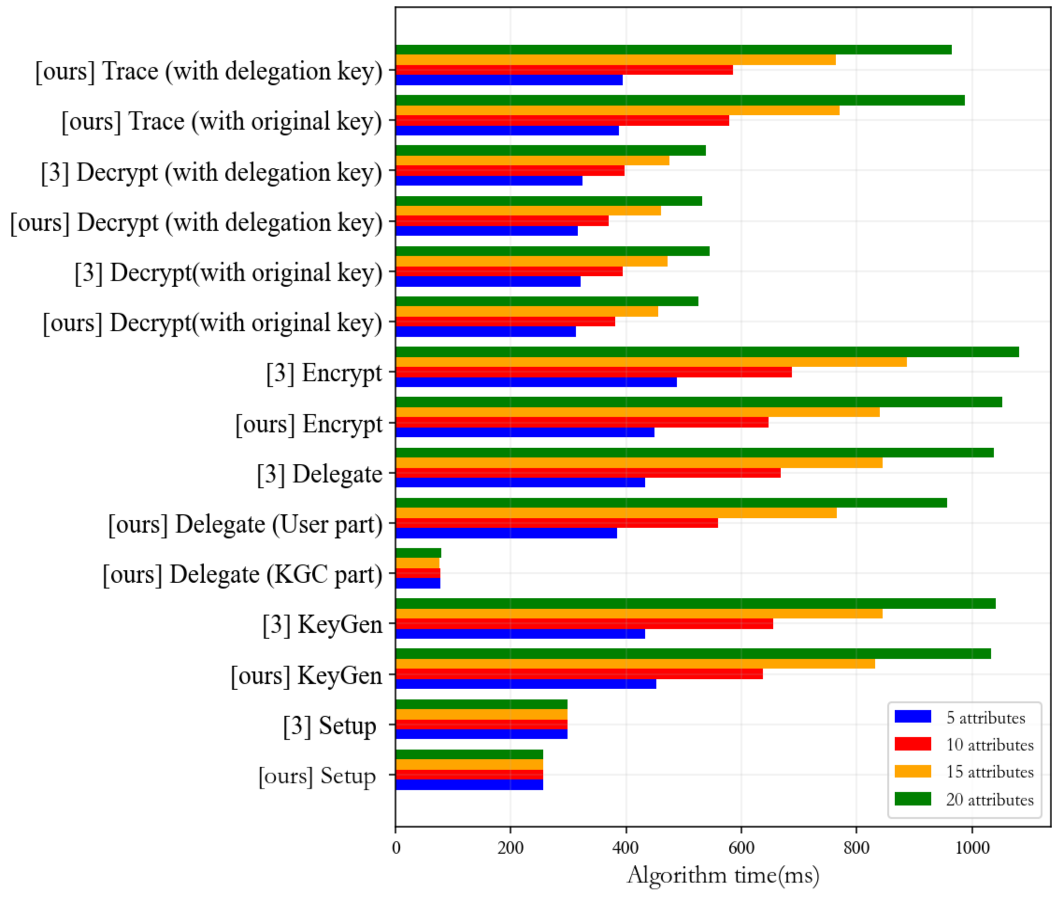

Fig. 6. Time cost of different algorithms with varying numbers of attributes on Linux Ubuntu

In the test, for the different number of attributes, we kept the ***Setup*** condition unchanged. The time cost of other algorithms rises with the increase in the number of attributes. We use the "*AND*" gate to define the policy to force both the delegator and the delegatee to use all attributes in the decryption process. Hence, the time cost for decryption both by the original key and the delegation key reflects the upper bound. The time cost of the KGC in the key delegation part is significantly less than its time cost for generating an original key. This shows that the key delegation scheme can enable the system to accommodate more users while reducing the workload of the KGC. We set up the attributes associated with the delegation key, which are the same as those of the original key, to reflect the upper bound of the time in ***Delegate*** (User part) and ***Trace*** .

In general, compared with [3], our scheme is slightly more efficient. The delegation key in [3] is generated independently by the KGC. It can be seen that the time cost for ***Delegate*** with the KGC in our scheme is much lower than that in [3]. As our scheme supports key traceability, the sum of the time cost of both ***Delegate*** algorithms (i.e., that of the KGC and that of the user) is higher than that in [3].

These results show that the key delegation mechanism can transfer the work of the KGC to the user, and let the user act as the authorization authority. This can not only make the whole system accommodate more users but also improve the efficiency of the system.

7. Conclusions

This article presents a fine-grained, traceable CP-ABE key delegation approach for the purpose of *read* permission delegation of outsourced data. In this study, based on the principle of least decryption ability comes from the principle of least privilege in access control, a minimum number of attributes corresponding to the access structures of certain ciphertexts are selected from the user’s attribute set. A delegation key is constructed by the minimal attribute set, and delegated to the delegatee to decrypt these ciphertexts. In this way, the generation time of the delegation key is the minimum, and the length of this key is the shortest. Controlling the delegatee's decryption ability through the delegation key provides flexibility and fine granularity of access control. An anti-collision feature prevents the delegatee from combining his ordinary key with the delegation key or combining several delegation keys that he possesses, to access data objects that he is not authorized to access. The KGC associates the user's identity with the key component so that the delegation key has traceable characteristics. Users complete the key delegation process under the participation and supervision of the KGC. The proposed approach supports multistage delegation, meaning that the delegatee can further delegate the key to other users. Furthermore, the time cost of the KGC to generate the delegation key is low, and it will not increase with an increase in the number of attributes. Therefore, our approach enables the system to accommodate more users to improve the efficiency of the system. The key delegation has practical implications. From the information sharing point of view, a user may not possess attributes associated with a delegation key, but he has the chance to access some data objects using the delegation key without having related attributes. Consequently, information sharing can be improved. From the scalability point of view, both original and delegation key owners can bring others into the system to access data. Additionally, permission delegation is one of the important issues in the access control scenario, and our approach enables key owners to delegate some of the *read* permissions to other users via key delegation.

However, our approach is not capable of controlling the width and depth of the key delegation. Black-box traceability is better than white-box traceability and it is more efficient and scalable. The white-box traceable CP-ABE scheme can only trace users possessing complete keys. It cannot trace users possessing incomplete keys. On the contrary, the black-box traceable CP-ABE can trace users possessing both complete and incomplete keys. In the future, we will investigate a more flexible and discretionary key delegation approach. A more discretionary but still secure key delegation scheme without the involvement of the KGC will reduce the computational cost of the KGC and communication costs between the KGC and users.

Appendix

A. Scheme analysis

Theorem 1 If the DBDH assumption holds, the security model in Section 3.2.2 of our scheme is no different from a plaintext attack.

Proof: The security model constructed in this study is similar to that in [25]. According to the security proof in [25], we can infer that our solution is safe.

Theorem 2 The security of traceability in our key delegation scheme is not weaker than [32].
Proof: Some key components of the user in our scheme have similar structures as in [32]. $D = g^{(\alpha+r)/(\beta+c)}$ contains components g^{α} and β of the master key. r is a random number

selected for the user, and a parameter c represents the user's identity information. Compared with [32], we construct D with a random number r, without any public parameter. It is difficult to obtain any one of $g^{\alpha/(\beta+c)}$ or $g^{r/(\beta+c)}$. According to the security proof of [32], it can be known that the traceability of the decryption key is safe. Therefore, D is also safe in our scheme.

Theorem 3 The security of the key delegation scheme in this study is not weaker than [32].

Proof: Compared with [32], in the key delegation algorithm of our scheme, the user obtains three parameters ($D^* = g^{(\alpha+r+r')/(\beta+c)} \| r' \| D'' = c'$) from the KGC so that the KGC can supervise the key delegation process, because it embeds the delegatee's identity into the key for tracing. The random number r' re-randomizes the key component. Therefore, the key delegation in this article is safe.

B. Test supplement

We measured the average time cost for algorithms in this scheme. The test environment was a PC with an Intel® Core™ i7-10510U CPU @ 1.80 GHz 2.30 GHz, with 16 GB RAM, and 64-bit Windows 10 (19042). The prototype system was developed in Java and run on Java Development Kit (JDK) 15. It was based on the JPBC (Java Pairing Based Cryptography) library 1.2.0 to implement bilinear operations. We chose 128-bit AES for the symmetric encryption algorithm. The degree of the polynomial used in the test was 20.

The number of attributes associated with the ciphertext, delegated key, and original key was the same for all algorithms, and keys met the ciphertext access structure.

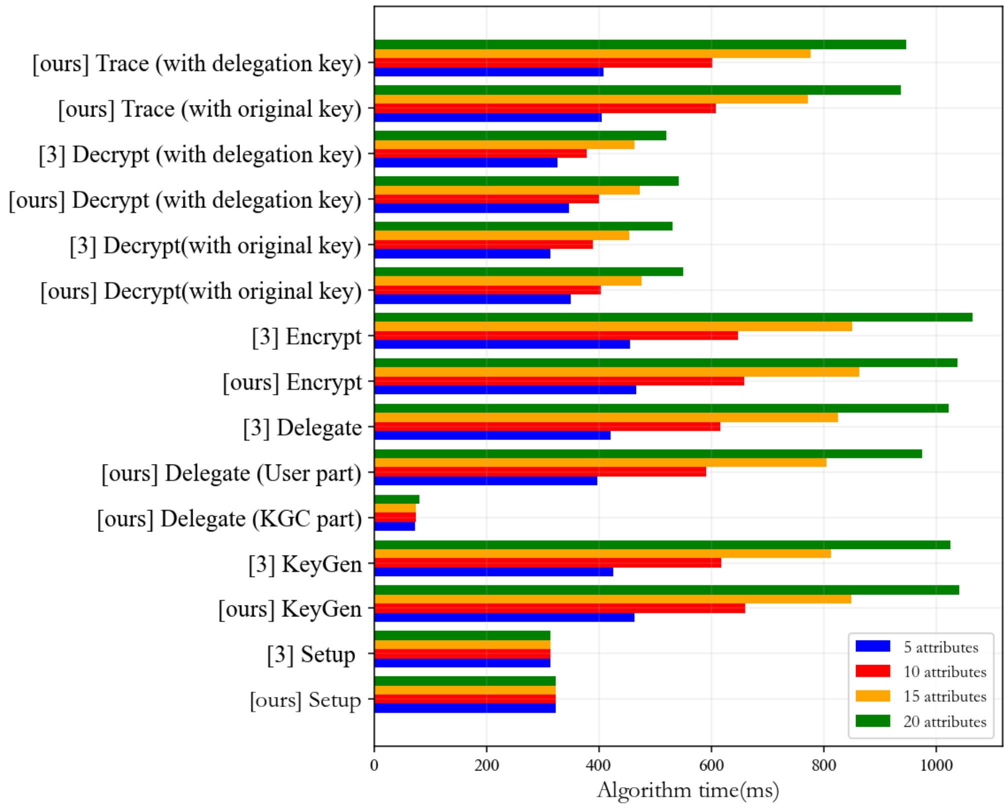

Fig. 7. Time cost of different algorithms with varying number of attributes in Windows 10

On the Windows 10 operating system, each algorithm is tested 10 times, and the variance is smaller than that obtained on Linux. We took the average of 10 tests, and the experimental results are shown in **Fig. 7.** The time costs for all algorithms on Windows are slightly higher than those on Linux Ubuntu.

References

[1] Sahai A., Waters B., "Fuzzy Identity-Based Encryption," in *Proc. of Cramer R. (eds) Advances in Cryptology – EUROCRYPT 2005. EUROCRYPT 2005. Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, vol. 3494, no. pp. 457-473, 2005. Article (CrossRef Link).

[2] Goyal, V., et al., "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," in *Proc. of ACM Conference on Computer and Communications Security*, pp.89-98. Oct. 2006. Article (CrossRef Link).

[3] Bethencourt J, Sahai A, Waters B., "Ciphertext-Policy Attribute-Based Encryption," in *Proc. of IEEE Symposium on Security & Privacy*, IEEE, pp.321-334. Article (CrossRef Link).

[4] L. Zhang, G. J. Ahn, B. T. Chu, "A Rule-Based Framework for Role-Based Delegation and Revocation," *ACM Transactions on Information and System Security*, vol. 6, no. 3, pp.404-441, August 2003. Article (CrossRef Link).

[5] Sandhu R, Coyne E, Feinstein H, et al., "Role-Based Access Control Models," *IEEE Computer*, vol. 2, no. 29, pp.38-47, 1996. Article (CrossRef Link).

[6] Barka E, Sandhu R., "Role-Based Delegation Model/ Hierarchical Roles (RBDM1)," in *Proc. of Computer Security Applications Conference*, IEEE, 2010. Article (CrossRef Link).

[7] Zhang, X., Oh, S., & Sandhu, R. S., "PBDM: a flexible delegation model in RBAC," in *Proc. of Symposium on Sacmat*, DBLP, June, pp. 149-157, 2003. Article (CrossRef Link).

[8] Schaad, A., "Detecting conflicts in a role-based delegation model," *Computer Security Applications Conference*, IEEE, 2001. Article (CrossRef Link).

[9] Crampton J, Khambhammettu H, "Delegation in role-based access control," *International Journal of Information Security*, vol. 7, pp. 123-136, 2008. Article (CrossRef Link).

[10] Ben-Ghorbel-Talbi M, Frédéric Cuppens, Cuppens-Boulahia N, et al., "A delegation model for extended RBAC," *International Journal of Information Security*, vol. 9, pp. 209-236, May. 2010. Article (CrossRef Link).

[11] Ghorbel-Talbi M B, Cuppens F, Cuppens-Boulahia N, et al., "Managing Delegation in Access Control Models," in *Proc. of 15th International Conference on Advanced Computing and Communications*, pp. 744-751, Dec. 2007. Article (CrossRef Link).

[12] Chun Ruan, Vijay Varadharajan, "Dynamic Delegation Framework for Role Based Access Control in Distributed Data Management Systems," *Distributed & Parallel Databases*, vol. 32, pp. 245-269, 2014. Article (CrossRef Link).

[13] Yan H, "A new role-to-role delegation model," in *Proc. of The 2nd International Conference on Information Science and Engineering*, pp. 1-4, Dec. 2010. Article (CrossRef Link).

[14] Wei Z, Jian S, Feng-Yu Y, et al., "Delegation Model Based on Delegation Depth and Role Range," *Computer Engineering*, vol. 36, no. 1, pp. 136-138, Jan. 2010. Article (CrossRef Link).

[15] Park S Y, Lee S H., "ID-Based Access Control and Authority Delegations," in *Proc. of Embedded and Ubiquitous Computing - EUC 2005 Workshops, EUC 2005 Workshops: UISW, NCUS, SecUbiq, USN, and TAUES*, Nagasaki, Japan, pp. 6-9, Dec. 2005. Article (CrossRef Link).

[16] Chunxiao Y E, Yunqing F U, et al., "Study on Delegation Revocation in Attribute Supported Delegation Model," *Computer Science*, vol. 6, no. 37, pp. 217-219, 2010. Article (CrossRef Link).

[17] Chunxiao Y, Zhongfu W, Yunqing F, et al., "An Attribute-Based Extended Delegation Model," *Journal of Computer Research and Development*, vol. 6, no. 43, pp. 1050-1057, 2006. Article (CrossRef Link).

[18] Y.-H. Wei, C.-E. Wang, M.-X. Ma, "Delegation authorization mechanism for workflow system," *Computer Integrated Manufacturing Systems*, vol. 1, no. 15, pp. 160-159, March. 2009.

[19] Liu Z, Cao Z, Wong D S, "White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Any Monotone Access Structures," *IEEE Transactions on Information Forensics & Security*, vol. 1, no. 8, pp. 76-88, Jan. 2013. Article (CrossRef Link).

[20] Ning J, Cao Z, Dong X, Wei L, Lin X, "Large Universe Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability," *Lecture Notes in Computer Science*, vol. 8713, pp.55-72, 2014. Article (CrossRef Link).

[21] Ning J, Cao Z, Dong X, et al., "White-Box Traceable CP-ABE for Cloud Storage Service: How to Catch People Leaking Their Access Credentials Effectively," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 883-897, Sept. 2018.Article (CrossRef Link).

[22] Wang S, Guo K, Zhang Y, "Traceable ciphertext-policy attribute-based encryption scheme with attribute level user revocation for cloud storage," *PLoS ONE*, vol. 13, no. 10, pp.1-23, Sept. 2018. Article (CrossRef Link).

[23] Yu G, Wang Y, Cao Z, Lin J, & Wang X, "Traceable and undeniable ciphertext-policy attribute-based encryption for cloud storage service," *International Journal of Distributed Sensor Networks*, vol. 15, no. 4, pp. 1-10, April. 2019. Article (CrossRef Link).

[24] Jiang Y, Susilo W, Mu Y, et al., "Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing," *Future Generation Computer Systems*, vol. 78, no. PT.2, pp. 720-729, Jan. 2018. Article (CrossRef Link).

[25] Yan Xixi, He Xu, Liu Tao, Ye Qing, Yu Jinxia, Tang Yongli, "Traceable attribute-based encryption scheme to resist key delegation abuse," *Journal of Communications*, vol. 41, no. 4, pp. 150-161, 2020. Article (CrossRef Link).

[26] Alibakhshikenari, M, Virdee, B. S, & Limiti, E, "Compact Single-Layer Traveling-Wave Antenna DesignUsing Metamaterial Transmission Lines," *Radio Science*, vol. 52, no. 12, pp. 1510-1521, Dec. 2017. Article (CrossRef Link).

[27] Alibakhshikenari M, Virdee, B. S, Ali A, and Limiti E., "Extended Aperture Miniature Antenna Based on CRLH Metamaterials for Wireless Communication Systems Operating Over UHF to C-Band," *Radio Science*, vol. 53, no. 2, pp.154-165, Jan. 2018. Article (CrossRef Link).

[28] Alibakhshikenari M, Babaeian F, Virdee B S, et al., "A Comprehensive Survey on 'Various Decoupling Mechanisms with Focus on Metamaterial and Metasurface Principles Applicable to SAR and MIMO Antenna Systems'," *IEEE Access*, vol. 8, pp. 192965-193004, Oct. 2020. Article (CrossRef Link).

[29] Alibakhshikenari, M., Virdee, B., Shukla, P., See, C., Abd-Alhameed, R., Khalily, M., Limiti, E., "Antenna Mutual Coupling Suppression Over Wideband Using Embedded Periphery Slot for Antenna Arrays," *Electronics*, vol. 7, no. 9, pp. 198-209, Sept. 2018. Article (CrossRef Link).

[30] Alibakhshikenari, Virdee, See, Abd-Alhameed, Falcone, and Limiti, "High-Isolation Leaky-Wave Array Antenna Based on CRLH-Metamaterial Implemented on SIW with ±30o Frequency Beam-Scanning Capability at Millimetre-Waves," *Electronics*, vol. 8, no. 6, pp. 642-657, Jun. 2019. Article (CrossRef Link).

[31] Alibakhshi-Kenari M, Naser-Moghadasi M, Sadeghzadeh R. A, Virdee B. S, and Limiti E, "A new planar broadband antenna based on meandered line loops for portable wireless communication devices," *Radio Science*, vol. 51, no. 7, pp.1109-1117, Jul. 2016. Article (CrossRef Link).

[32] Guan Z, Li J, Zhang Y, et al., "An efficient Traceable Access Control Scheme with Reliable Key Delegation in Mobile Cloud Computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, pp. 208, Sept. 2016. Article (CrossRef Link).

[33] Liang X, Cao Z, Lin H, & Shao J, "Attribute based proxy re-encryption with delegating capabilities," in *Proc. of International Symposium on Information, Computer, and Communications Security*, Shanghai Jiao Tong University, pp. 276-286, 2009. Article (CrossRef Link).

[34] Liu Z, Wong D.S., "Traceable CP-ABE on Prime Order Groups: Fully Secure and Fully Collusion-Resistant Blackbox Traceable," *Qing S, Okamoto E, Kim K, Liu D. (eds) Information and Communications Security. ICICS 2015. Lecture Notes in Computer Science*, vol. 9543, pp. 109-124, March. 2016. Article (CrossRef Link).

[35] Qiao H, Ba H, Zhou H, et al., "Practical, Provably Secure, and Black-Box Traceable CP-ABE for Cryptographic Cloud Storage," *Symmetry*, vol. 10, no. 10, pp. 482-499, Oct. 2018. Article (CrossRef Link).
[36] Zhang, W, Wu, Y, Zhang, Z, Xiong, H, & Qin, Z, "Multi-Authority Ciphertext-Policy Attribute Based Encryption with Accountability," *ArXiv*, vol. abs/2009.04748, Sept. 2020. Article (CrossRef Link).
[37] Li, H, Deng, L, Yang, C, & Liu, J, "An enhanced media ciphertext-policy attribute-based encryption algorithm on media cloud," *International Journal of Distributed Sensor Networks*, vol. 16, no. 2,pp. 1-15, Feb. 2020 Article (CrossRef Link).
[38] Shamir A, "Identity-Based Cryptosystems and Signature Schemes," *Crypto'84*, Berlin, Heidelberg, Springer, vol.196, pp. 47-53, 1984. Article (CrossRef Link).

Jiajie Du is currently a graduate student in College of Mathematics and System Science, Xinjiang University, China. Her research mainly focuses on information security and cryptography.

Nurmamat Helil received his B.S, M.S, and Ph.D. degrees in School of Mathematical Sciences, Peking University in 2000, 2003, and 2008, respectively. He is a full Professor of College of Mathematics and System Science, Xinjiang University, China. From April 2010 to April 2011, he worked as a post-doctor at the School of Computer Science and Engineering, Chung-Ang University, Korea. From April 2016 to April 2017, he worked as a Visiting Research Scholar in the Department of Computer Science and Engineering, University of Minnesota Twin Cities, USA. His research interests include information system security, access control, and cloud storage security.