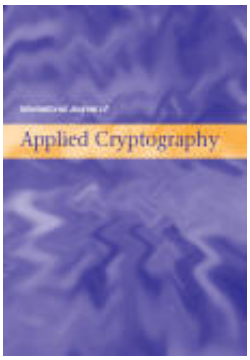
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research




**International Journal of Applied Cryptography**

ISSN online: 1753-0571 - ISSN print: 1753-0563
https://www.inderscience.com/ijact

# PPADMA-ABE: a novel privacy-preserving and auditable attribute-based encryption under dynamic multi-authority setting

Zhifa Deng, Jiageng Chen, Shixiong Yao, Pei Li

**DOI:** 10.1504/IJACT.2023.10061466

**Article History:**

| | |
|---|---|
| Received: | 12 February 2023 |
| Last revised: | 19 March 2023 |
| Accepted: | 30 March 2023 |
| Published online: | 03 May 2024 |

# PPADMA-ABE: a novel privacy-preserving and auditable attribute-based encryption under dynamic multi-authority setting

Zhifa Deng, Jiageng Chen*, Shixiong Yao and Pei Li

School of Computer Science,
Central China Normal University,
Wuhan, China
Email: zhifadeng@mails.ccnu.edu.cn
Email: jiageng.chen@ccnu.edu.cn
Email: yaosx@ccnu.edu.cn
Email: peili@ccnu.edu.cn
*Corresponding author

**Abstract:** Attribute-based encryption (ABE) enables a flexible approach to data storage in the cloud by allowing many users to encrypt data with attributes and ensuring that only authorised users with the matching attributes can access the data. However, the original ABE system is often static, which limits its flexibility. Therefore, we propose a novel attribute-based encryption with security auditing, dynamic multi-authority, and privacy-preserving (PPADMA-ABE) system that enhances the practicality and flexibility. Our scheme supports dynamic changes under the coexistence of multiple authorities and enables self-auditing to meet potential practical requirements. Additionally, we have outsourced a significant amount of user-side operations, which reduces the decryption cost at the terminal to a constant level. Finally, we prove that our scheme is secure against selective chosen-ciphertext attacks and can withstand collusion by malicious users or authorities. In summary, PPADMA-ABE provides a more practical and flexible solution for dynamic multi-authority ABE with privacy preservation.

**Keywords:** multi-authority ABE; dynamicity; outsourcing; auditing; collusion resistance; privacy preserving.

**Reference** to this paper should be made as follows: Deng, Z., Chen, J., Yao, S. and Li, P. (2023) 'PPADMA-ABE: a novel privacy-preserving and auditable attribute-based encryption under dynamic multi-authority setting', *Int. J. Applied Cryptography*, Vol. 4, Nos. 3/4, pp.176–194.

**Biographical notes:** Zhifa Deng is a graduate student majoring in Computer Science and Technology at the School of Computer Science, the Central China Normal University, focuses academically on attribute-based encryption, cryptographic protocol security, blockchain network technology, and related areas.

Jiageng Chen received his BS from the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST) in 2004, and his MS and PhD from the School of Information Science, JAIST in 2007 and 2012, respectively. He worked as an Assistant Professor in the School of Information Science, JAIST from 2012 to 2015. He is an Associate Professor at the School of Computer, Central China Normal University, and also an associate editor of the *Journal of Information Security and Application*. His research areas include cryptography, especially in privacy-preserving computing, blockchain technology, etc.

Shixiong Yao received his PhD in Information Security in the School of Cyber Science and Engineering in Wuhan University. He is currently a Lecturer with the School of Computer, Central China Normal University, Wuhan. He has published papers in many international journals and conferences, such as the INFOCOM, TrustCom, Connection Science, FGCS and the *International Journal of Network Management*. His research interests are in the areas of blockchain and identity management.

Pei Li received his BS in Optoelectronics Engineering from the Huazhong University of Science and Technology (HUST) in 2010, his MS in Computer Science from the University of Paris Sud in 2012 and his PhD of Computer Science from the University of Bordeaux in 2016. During 2015–2016, he stayed at Pierre and Marie Curie University as visiting scholar. Currently, he is a Lecturer at the School of Computer, Central China Normal University. His research interests are heterogeneous computing, code optimisation and information security.

This paper is a revised and expanded version of a paper entitled 'An outsourced multi-authority attribute-based encryption for privacy protection with dynamicity and audit' presented at EAI BlockTEA 2022 – 2nd EAI International Conference on Blockchain Technology and Emerging Applications, Copenhagen, Denmark, 21–22 November 2022.

## 1 Introduction

With the increased awareness of personal privacy protection, cloud computing and related information technology have become heavily applied in various scenarios to safeguard users' privacy. Cloud computing has emerged as a new and popular business paradigm, and organisations and individuals are storing large amounts of data in the cloud, rather than to build and maintain the local data centres, which can greatly reduce the operating cost. However, the data stored in the cloud server often contains sensitive information. Traditionally, data servers were trusted to keep data confidential, and access control policies were assumed to be executed correctly. However, this technology is no longer applicable in the era of big data, where cloud service providers (CSP) and end-users do not belong to the same trusted domain, and the CSP may not be completely trusted, since the hardware platform is not directly controlled by the data owner. Therefore, it is crucial to protect privacy and security when interacting with others. Cryptographers are actively exploring ways to reduce users' concerns about data privacy. An effective way is to encrypt the data before uploading to the server, thus protect the data from attackers whose target is the cloud server. However, the encrypted data also needs to be shared, and public-key encryption or symmetric encryption do not offer flexible access control. A better solution to this problem is attribute-based encryption (ABE), which was first proposed by Sahai (2005). ABE aims to protect the confidentiality of sensitive data and is classified into CP-ABE and KP-ABE according to where the policy is embedded (Bethencourt et al., 2007). Both of them can prevent users who do not meet the policy requirements from accessing the data, even if they collude.

In an ABE system, the encryptor does not know who can decrypt the ciphertext, and the receiver does not know who the encryptor is. This provides privacy protection for various participants, enabling them to exchange private information anonymously to a certain extent. CP-ABE is mostly designed for fine-grained data access control in distributed platforms, making it attractive in the field of cloud data storage control. However, with the further development of the internet and explosive growth of data, users' requirements for the system have become increasingly complex and dynamic. In the past, ABE was mostly in a static state, with relevant system parameters fixed at the beginning, limiting the flexibility of the system. Previous works such as Liu et al. (2018), Ling et al. (2021) and Deng (2014) have addressed only partial updates of the policy or the user, which cannot handle dynamic changes and meet the growing potential needs of users.

In addition, the high cost of encryption and decryption is also a serious bottleneck for ABE, which may essentially prevent its widespread deployment, especially on resource-constrained devices or platforms. Most existing pairing-based CP-ABE systems have linearly dependent decryption overhead, which often increases with the complexity of the access policy, making it a challenging issue. Outsourcing decryption is an effective way to reduce the decryption overhead of users by outsourcing a large number of decryption operations to CSP. However, since CSPs are not fully trusted third parties, verifying the correctness of outsourced decryption is a tricky problem. Previous solutions such as those proposed in Green et al. (2011) and Zhang et al. (2015) only focused on the efficiency or security of the outsourcing process without considering the verifiability of outsourced decryption, which is crucial in an untrusted cloud environment. While system efficiency can benefit from outsourcing, it comes at the cost of security compromise to untrusted third-party services. In a word, none of the above-mentioned research has focused on combining dynamicity and auditable mechanisms under the multi-authority setting.

### *1.1 Motivation*

The previous background motivates us in the following aspects:

1 The existing scenarios have high requirements for dynamicity. It inspires us to combine the multi-authority with the dynamic feature to realise the flexible management of the system.

2 How to safely offload the 'expensive' linear pairing computation operations in the decryption process to the cloud.

3 How to efficiently check the validity of the decryption results returned by the cloud.

4 Since the system members are dynamically changing and the number of attributes is increasing, how to effectively resist the collusion among users or authorities.

In summary, none of the previous researches has focused on combining dynamicity and auditable mechanisms in the multi-authority setting. In this paper, we are motivated by the dynamic policy updating in Ling et al. (2021), multi-authority attribute encryption scheme in Chase and Chow (2009), outsourcing decryption in Zhang et al. (2015), as well as several other techniques in Han et al. (2015), Waters (2008), Kan and Jia (2012) and Ning et al.

(2017), and propose a more flexible ABE system that focuses on user privacy protection and security in the following aspects:

- We propose that the number of attribute authorities should be set by multi-authority while maintaining dynamicity, so as to achieve one-to-one correspondence between attributes and authorities and ensure the system's dynamicity. Thus, solve the system bottlenecks and improve system efficiency and flexibility.
- Due to the large computational overhead in the decryption phase in most of the schemes (Green et al., 2011; Zhang et al., 2015; Armknecht et al., 2014; Yang et al., 2013; Li et al., 2014), we outsource the related parts to the cloud server.
- The system will not introduce an additional third party to act as the auditor, but instead we will distribute the role within the system. In this way, the decryption results of the outsourced decryption cloud server can also be verified (Ning et al., 2017) and strictly ensure the decryption security.
- Regarding the security and privacy protection, we are inspired by Rahulamathavan et al. (2016) to change the ciphertext form to handle collusion problem.

Therefore, we investigate the above interesting issues and further propose a security auditing and dynamic multi-authority ABE scheme with privacy protection.

### *1.2 Our contribution*

To the best of our knowledge, our proposal is the first instance that combines multi-authority with dynamic features. The main contributions of this work can be summarised as follows:

- *Dynamic change in the multi-authority setting:* We have implemented several dynamic operations in our proposed system, such as the dynamic join, withdraw, and update of attribute authorities, as well as the dynamic join, withdraw, and update of users. Additionally, we have also enabled dynamic updates of various attributes in the policy. There are multiple authorities working together in the system, which can make the system more stable. Each attribute authority is responsible for attribute management, and the CA is responsible for managing each AA and notifying the current status of each one through a broadcast mechanism. The presence of multiple authorities in the system improves its stability while still ensuring dynamic performance.
- *Auditability of decryption by AA and CA:* The dual audit mechanism we proposed can effectively ensure the integrity and correctness of the outsourced decryption without relying on a third party auditor. This mechanism is designed to be lightweight and efficient, without introducing any additional parameters or computational costs to the user. The decryption information will be audited twice by the attribute authorities (AA) and the central authority (CA) after the decryption request is executed. Therefore, we can detect any attempts of malicious behaviour and verify the correctness of the decryption results. Compared to verifiable methods described in Ma et al. (2015) and Hui et al. (2017), our auditable method does not require any modifications to the ciphertext of our system and does not add any computational overhead to the user.
- *Effective resistance to collusion:* The scheme restricts the collusion between malicious users and malicious authorities from the key distribution and decryption elements, and provides detailed proof to optimise the third party disclosure collusion existing in the previous scheme.

### *1.3 Paper organisation*

The rest of the study is organised as follows. In Section 2, we review a series of related works of ABE, compare and analyse several types of attribute encryption with different features. In Section 3, we introduce the preliminaries including the related definitions and complexity assumptions used in the scheme. In Section 4, we present the system model, scheme framework, security model and security assumptions. In Section 5, we give the concrete scheme construction. The security analysis of our scheme is presented in Section 6. In Section 7, an evaluation and comparison of relevant performance is provided. Finally, we conclude our paper in Section 8.

## 2 Related work

In 1984, Shamir proposed the concept of identity-based encryption (IBE), which uses identity information as the public key to encrypt messages. In 2001, Boneh proposed a provably secure IBE scheme based on Weil pairing, and enhanced it by the technique from Fujisaki and Okamoto (1999). In 2005, Sahai proposed a new type of IBE, the fuzzy IBE, which is the prototype of ABE, but it only allowed fixed identities in the system, which seems to be closer to the access control mechanism. We can view IBE as a very special case of ABE. From then on, various ABE schemes have been gradually proposed. According to the number of authority, there are single authority, multi-authority and de-centred multi-authority schemes. Single-authority and multi-authority settings have been proposed to address the management issues in both encryption schemes. According to the characteristics of ABE, it can be divided into revocation ABE, key delegate encryption, reproxy, audit accountability, outsourcing, online and offline, and lattice-based attribute encryption. Different schemes have different properties, which makes ABE have stronger development potential.

In a single-authority ABE, both attribute management and key distribution are handled by a single authority. However, in most practical scenarios, users have more than one attribute, then a multi-authority attribute-based encryption (MA-ABE) system is proposed. In a multi-authority ABE system, different attribute authorities manage different attribute sets and distribute corresponding attribute keys. In 2007, Chase implemented the MA-ABE scheme for the first time (Chase, 2007). In Chase (2007), the attribute authority (AA) are independent of each other and do not need to exchange information, but it needs a fully trusted central authority (CA) to manage all private information. What's more, it achieves collusion resistance by assigning a unique global identifier (GID) to each user. Later, Lewko and Waters (2011) proposed a new MA-ABE scheme called decentralising CP-ABE scheme, in which the authority centre CA is removed, and any party can become an authority.

Moreover, due to the complex pairing and exponentiation operations in ABE, outsourcing complex operations to cloud server has become an effective solution. The concept of outsourcing retrievable proof is proposed by Armknecht et al. (2014), in which users can entrust external auditors to execute and verify POR with cloud providers. Outsourcing decryption is introduced into the ABE system by Green et al. (2011), so that the complex operations in the decryption stage can be outsourced to the cloud server, leaving only an exponentiation operation for the user to recover the plaintext. Although they entrusted complex operations to the cloud server, they did not consider the correctness of the results returned by the cloud. In order to solve the above problem, Ren et al. (2015) proposed a mutual verifiable provable data auditing method. Later, Lai et al. (2013) introduced the verifiability of ABE to verify the correctness of outsourced decryption. Zhang et al. (2015) revisited verifiable outsourced ABE (Lai et al., 2013) and proposed a more efficient approach to construct. In addition to outsourcing decryption, Li et al. (2014) and Hui et al. (2017) also considered key distribution during outsourcing. Green et al. (2011) proposed to outsource decryption by transferring computationally expensive operations on ciphertexts (such as bilinear pairing operations) to devices with more powerful computing power, thereby effectively reducing the computational cost of users. In 2020, Sethi et al. transferred the expensive decryption operation to a device with stronger computing power, thereby reducing the computing overhead essentially. In addition, they also obtained the partial ciphertext, so that it was unnecessary to obtain all the ciphertext for decryption, further integrating the overhead resources to improve the computing performance of the system.

Also, after the multi-authority attribute encryption is proposed, a series of functional features are also derived, such as policy updating or hiding (Liu et al., 2018), attribute revocation (Yang et al., 2013), searchable ABE (Green et al., 2011), ABE on lattices (Liu et al., 2017b), online or offline combined computing (Hohenberger and Waters, 2014), large universe mechanisms (Lian et al., 2019), outsourced scheme (Zhang et al., 2015), verifiable scheme (Ma et al., 2015), and so on. However, only some papers are aimed at member management or auditing or outsourcing (Yu et al., 2010; Ren et al., 2015; Zhang et al., 2015), just pay attention to one aspect, they cannot use multi-authority decentralised management while ensuring dynamicity, and at the same time have to take into account decryption outsourcing and audit the correctness of decryption results. None of the above-mentioned research has focused on combining dynamicity and auditable mechanisms with multi-authority.

## 3 Preliminaries

### 3.1 Access structures

*Definition 3.1 [access structure (Shamir, 1979)]:* Let $\{P_1, P_2, ..., P_n\}$ be a set of parties. A collection $A \subseteq 2^{\{P_1,P_2,...,P_n\}}$ is monotone if $\forall B, C$: if $B \in A$ and $B \subseteq C$ then $C \in A$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection). A of non-empty subsets of $\{P_1, P_2, ..., P_n\}$, i.e., $A \subseteq 2^{\{P_1,P_2,...,P_n\}} \backslash \{\emptyset\}$. The sets in $A$ are called the authorised sets, and the sets not in $A$ are called the unauthorised sets.

In the subsequent discussions, the set of attributes in ciphertext will be embedded in access policy A, so we limit our attention to the monotonous access structure, but the 'not' logic can also be included at the cost of doubling the number of attributes in the system, resulting in a more general access structure. From now on, access structures in this article refer to monotone access structures unless otherwise stated.

### 3.2 Lagrange interpolation

According to Beimel (1996) and Pohoata (2008), a Lagrange interpolating polynomial is a polynomial of degree not greater than $(n-1)$ that passes through $n$ points $(x_i, y_i), ..., (x_n, y_n)$, and is given by,

$$p(x) = \sum_{j=1}^{n} p_j(x) \tag{1}$$

where

$$p_j(x) = y_j \prod_{k=i,...,n,k\neq j} \frac{x - x_k}{x_j - x_k} \tag{2}$$

For $i \in Z$ and $S \subseteq Z$, the Lagrange coefficient $\triangle_{i,s}(x)$ is defined as

$$\triangle_{i,s}(x) = \prod_{\forall j \in S, j\neq j} \frac{x - j}{i - j} \tag{3}$$

### 3.3 Bilinear pairings

Let $G$ and $G_T$ be two multiplicative cyclic groups of prime order $p$. Let $g$ be a generator of $G$ and $e : G \times G \to G_T$ be a bilinear map with the properties:

- bilinearity: $e(A^x, B^y) = e(A, B)^{xy}$ for all $A, B \in G$ and $x, y \in Z_p$
- non-degeneracy: $\exists A \in G_1, B \in G_2, e(A, B) \neq 1$, where 1 is the identity of $G_T$
- efficient computability: there exits an algorithm that can efficiently compute $e(A, B)$ for all $A \in G_1, B \in G_2$.

### 3.4 The decisional bilinear Diffie-Hellman problem

Let $G_1, G_2$ and $G_T$ be three cyclic groups of prime order $q$, $P$ and $Q$ be arbitrarily-chosen generators of $G_1$ and $G_2$, respectively, and $e : G_1 \times G_2 \to G_T$ be a bilinear mapping. Given $(P, Q, aP, bP, cP, aQ, bQ, cQ, Z)$ for some $a, b, c \in Z_q^*$ and $Z \in G_T$, decide if $Z = e(P, Q)^{abc}$.

An algorithm $B$ that outputs $b' \in \{0, 1\}$ has advantage $\epsilon$ in solving the DBDH problem if

$$|Pr[B(P, Q, aP, bP, cP, aQ, bQ, cQ, e(P, Q)^{abc})] - Pr[B(P, Q, aP, bP, cP, aQ, bQ, cQ, Z)]| \geq \epsilon \quad (4)$$

*Definition 3.2 (the decisional-BDH assumption):* We say that the decisional BDH assumption holds if no polynomial-time adversary has non-negligible advantage in solving the decisional BDH problem (Boneh, 2001).

### 3.5 Trapdoor function

We say a trapdoor function is an algorithm that can be effectively reversed when you know the trapdoor; However, it is very difficult to calculate reversals without knowing the trapdoor.

*Definition 3.3 [the trapdoor function scheme (Boneh and Shoup, 2020)]:* Let $X$ and $Y$ be finite sets. A trapdoor function scheme $T$, define over $(X, Y)$, is a triple of algorithm $(G, F, I)$, where

1 $G$ is a probabilistic key generation algorithm that is invoked as $(pk, sk) \xleftarrow{R} G()$, where $pk$ is called a public key and $sk$ is called a secret key.

2 $F$ is a deterministic algorithm that is invoked as $y \leftarrow F(pk, x)$, where $pk$ is a public key (as output by $G$) and $x$ lies in $X$. The output $y$ is an element of $Y$.

3 $I$ is a deterministic algorithm that is invoked as $x \leftarrow I(sk, y)$, where $sk$ is a secret key (as output by $G$) and $y$ lies in $Y$. The output $x$ is an element of $X$.

Also, $sk$ is the trapdoor of this function. If the $sk$ is obtained, it can be substituted into the algorithm to calculate the function value, i.e., we have $I(sk, F(pk, x)) = x$.

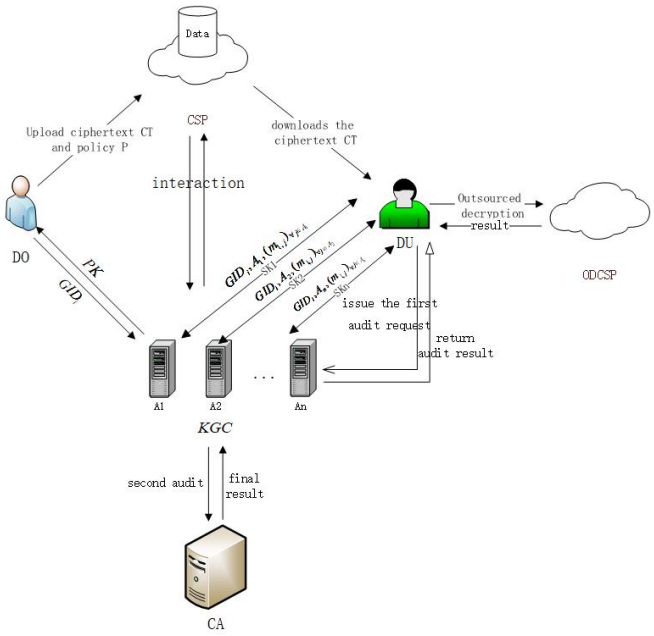
## 4 Our construction

### 4.1 System model

Our system consists of central authority (CA), attribute authority (AA), data owners (DO), data users (DU), cloud service provider (CSP) and outsourcing decryption cloud service providers (OD-CSP). The system architecture of our work is shown in Figure 1.

- *Central authority (CA):* In our scheme, the CA plays a critical role in establishing and maintaining the system, as well as processing dynamic requests from other entities like DU and AA. The CA is fully trusted and responsible for managing all AA operations, including AA registration, revocation, and updating, as well as user enrolment. However, it is not involved in attribute management. In addition to its administrative roles, the CA also acts as an auditor to ensure the correctness of outsourced decryption results. Since this auditing process can be costly, we delegate the preliminary auditing to the AA and let the CA act as a secondary auditor. This approach helps to improve the efficiency of the system, as the cost of auditing is mainly due to symmetric decryption.
- *Attribute authority (AA):* The AA is an entity responsible for handling different attributes. They exist independently of each other, managing different attributes and issuing corresponding decryption keys. The AA receives requests for GID from legitimate users and issues decryption keys to them based on their attributes. They can also generate update keys, which are used to update the keys of users in case some attributes of some users are added or revoked. In addition, each AA is responsible for auditing the content decrypted by the ODCSP and verifying its correctness.
- *Data owner (DO):* The DO defines data access policies using ABE on the specific encrypted content, and then encrypts the data through the policy. The access policy specifies the attributes required to access the ciphertext data, allowing for fine-grained data access control, and determines which users are able to access it.
- *Data user (DU):* The DU is an entity that wants to access the data and decrypt it. Once registered in the system, each user will have a unique GID. When a user has a ciphertext that needs to be decrypted, he/she sends its GID to the authority. Note that the authority does not get any information about the user attributes. After the authority verifies that the GID is correct, if the user has a set of attributes that satisfy the encrypted data access policy defined by DO, then he/she will use the attribute set for decryption to obtain the ciphertext content.

**Figure 1** The overall design of system (see online version for colours)




- *Cloud service provider (CSP):* The cloud service provider (CSP) is an entity responsible for storing the data. The DO submits the encrypted ciphertext data to the CSP for storage, which makes it convenient for DU to retrieve the ciphertext in the later stage. Meanwhile, if an AA wants to access the ciphertext during an auditing stage, it will interact with the CSP to verify the integrity and consistency of the ciphertext data during the auditing process.
- *Outsourced decryption cloud service provider (OD-CSP):* The OD-CSP is an entity responsible for decrypting the ciphertext sent by the user and calculating the partial session key. The DU obtains the ciphertext from CSP and obtains the partial session key from AA with the help of ODCSP, so ODCSP also plays an extremely important role. After receiving the complete and necessary information, ODCSP performs the fast calculation of bilinear pairings, and completes the whole operation with the cooperation of AA, so as to reduce the overhead of the whole system. The algorithm improves the decryption efficiency by reducing the number of pairing and exponentiation operations required by users.

### 4.2 The scheme framework

Based on the system model described above, our scheme consists of the following algorithms: system initialisation (CASetup(), AAEnrol(), UserEnrol()), execution (KeyGen(), Update(), Revoke()), encryption and decryption (OD-CSP), audit (CA and AA), decryption (user).

**Table 1** Notations

| *Notation* | *Meaning* |
|---|---|
| $\lambda$ | A security parameter |
| $S_{AA}$ | Set of all attribute authorities (AA) |
| $S_{A_i}$ | Set of attributes maintained by $AA_i$ |
| $S_{u_{id}}$ | Set of attributes owned by user $u_{id}$ |
| $U$ | Set of registered users |
| $Attr(N)$ | Attribute associated with node $N$ |
| $val(N)$ | Attribute value associated with node $N$ |
| $N_L$ | Set of all leaf nodes of access tree $T$ |
| $k_N$ | Threshold value associated with node $N$ |
| $d_N$ | Degree of the polynomial of node $N$ |
| $parent_{(N)}$ | Parent node of node $N$ |
| $index_{(N)}$ | Index associated with node $N$ |

First, some symbol notations in this chapter are described in Table 1. Then, our scheme is specifically constructed as follows.

#### 4.2.1 System initialisation

- $CASetup(1^\lambda) \rightarrow ((CA_{sk}, CA_{pk}), (CA_{sig}, CA_{verify}))$: The algorithm takes the security parameter $\lambda$ as input to generate the $PK$ and $SK$ of the CA, as well as the signature and its verification key pair $(CA_{sig}, CA_{verify})$.
- $AAEnrol(CA_{sk}, CA_{pk}, AA_{\text{info}}) \rightarrow (AA_i, AA_{i,sk}, AA_{i,pk})$: In this algorithm, the CA issues a unique ID $AA_{id}$ to each attribute authority. And then generate the public key($AA_{i,pk}$) and secret key($AA_{i,sk}$) for each of $i^{\text{th}}$ authority.
- $UserEnrol(CA_{sk}, CA_{pk}, u_{info}) \rightarrow (u_{id}, cert(u_{id}), u_{sk}, u_{pk})$: The algorithm makes use of the public and private key pair of CA $(CA_{sk}, CA_{pk})$ and user information $(u_{info})$ to assigns a unique global user's identity $u_{id}$ (here, we use the method in Chase and Chow (2009) to obtain the GID, so that the user can interact with other entities without revealing private information) and a certificate cert($u_{id}$) for the user. At the same time it generates the user's public and private key pair($u_{pk}, u_{sk}$).

#### 4.2.2 Execution

- $KeyGen(CA_{pk}, AA_{i,sk}, u_{id}, S_{u_{id}}) \rightarrow (SK_{u_{id}})$: This algorithm generates a decryption key for the user. Before the algorithm is executed, the system will verify the validity of the user. If the user is valid, the system enters the CA's public key, the current user's GID and the private key for $i^{\text{th}}$ AA, user attribute set $S$, then generates the decryption key for the outsourcing server to decrypt.
- $Update(u_{id}, j, m) \rightarrow (SK_{u_{id}, j \in S_{u_{id}}})$: When the user $u_{id}$ needs to change his/her attribute value, then the algorithm is executed to update it to a new value $m'_{id,j}$. The new value can either exist in the system

or be newly added, so as to meet the user's demand for changing the attribute value in the system.

- $Revoke(u_{id})$: When the $u_{id}$ user wants to leave the system, then the algorithm will be executed and the certificate will be revoked. What's more, the credentials of system will also be updated.

#### 4.2.3 Encryption and decryption

- $Encrypt(CA_{pk}, AA_{i,pk}, T, K, M) \rightarrow (CT)$: This algorithm takes public parameters of CA and the authority of the managed attributes, an access tree structure $T$, a key $K$, and a message $M$. The algorithm output the result of symmetric encryption under $K$, which is ciphertext $CT$.
- $Decrypt_{ODCSP}(CT, SK_{u_{id}}) \rightarrow (K_1)$: DU initiates an outsourced decryption calculation request. The algorithm is executed by the ODCSP. After receiving the ciphertext and decryption key from DU, output the partial session key $K_1$.

#### 4.2.4 Audit

- $Audit_{AA}(AA_{i,sk}, CT, Decrypt_{ODCSP}(CT, SK_{u_{id}})) \rightarrow$ (1/0): The AA performs the preliminary audit. The algorithm inputs the secret key $AA_{i,sk}$, the ciphertext CT, and the partial session key $K_1$ obtained from the ODCSP. Then the AA calculates the one-way trapdoor function by using its private key to obtains the XOR result of the function value and the partial session key $K_1$. Finally, the calculated result is compared with $\bar{M}$ in the ciphertext. If they are consistent, the algorithm outputs 1; otherwise returns 0.
- $Audit_{CA}(Audit_{AA}(*) == 1, CA_{pk}, CT, K, M) \rightarrow$ (1/0): The CA performs a secondary auditing. Since the AA is not fully trusted, a trusted CA is required to conduct a secondary auditing on the results of AA to ensure the correctness. The algorithm takes the CA's public key, the ciphertext CT, and the session key $K$ (since CA is absolutely trusted, the CA can obtain the partial session key $K_2$ of the user. Combined with the partial session key $K_1$ of the ODCSP, the complete session key $K$ is obtained). What's more, the algorithm will judge the result returned by the $Audit_{AA}$, if it returns 1, the algorithm can be successfully executed; otherwise, it will fall back to the previous step. Lastly, it is compared with the initial encrypted M, if $D_K(\bar{M}) = M$ then return 1, otherwise return 0.

#### 4.2.5 Decryption (user)

- $Decrypt_{DU}(Audit_{CA}(*) == 1, cert(u_{id}), CT, Decrypt_{ODCSP}(CT, SK_{u_{id}}), K_2) \rightarrow (M/\perp)$: The algorithm will judges the result returned by the $Audit_{CA}$, if it returns 1, the algorithm can be successfully executed; otherwise, it will fall back to the previous step. The algorithm is performed by the DU, if $Audit_{CA}(CA_{pk}, CT, K, \bar{M}, M) \rightarrow 1$, the algorithm (run by DU) outputs the message M. Otherwise, it outputs $\perp$.

### 4.3 Security assumptions and requirements

In the following analysis, we will rely the following assumptions:

- The CA must be completely trusted, and there must be no collusion by any other party.
- The AA is honest but curious, not completely trusted, and may collude with each other.
- The DU is honest but curious, and may collude with other unauthorised users out of the temptation of profit.
- CSP and OD-CSP are honest but curious, just like DU. They will actively and faithfully perform their duties, but may also be curious about the data content.

Also, we make some requirements as follows:

- Data confidentiality: Unauthorised users who do not conform to the access policy attributes are not allowed to access the data plaintext, and illegal users and the possibility of collusion are blocked.
- Backward and forward secrecy: In the practical scenario of ABE, the backward secrecy means that any user with an attribute (satisfying the access policy) has no right to access the plaintext before owning the attribute; and the forward secrecy means that any user whose attribute is revoked or has left the system can no longer access the plaintext data, unless the revocable attribute he holds satisfies other valid access policies.
- Collusion-resistance: If multiple users are going to collude, they cannot decrypt a ciphertext by combining their attributes even if they cannot decrypt it alone. Since we assume that CSP and ODCSP are not necessarily fully trusted, we do not consider active attacks from revoked users by colluding with them as in Yu et al. (2010).

### 4.4 Security model

Here, we describe a selectively CP-ABE and chosen-ciphertext attack (sCP-IND-CCA) model for the PPADMA-ABE scheme, played by a game between a challenger and an adversary. The model is called selective because it allows the adversary to adaptively choose the access control tree $T^*$ at the beginning of the game. Let $A$ denote the adversary who attempts to attack the scheme, $C$ denote the challenger who encrypt the challenge ciphertext. Formally, this is represented by the following game interaction between $A$ and $C$.

- *Init:* At the beginning of the game, the adversary A can choose which access control policy $T^*$ he wants to challenge and sends it to the challenger $C$.
- *Setup:* The challenger runs the system initialisation algorithm to establish the CA of the system and complete the creation and setup of the AA. Next, the challenger exploits the system algorithm to generate the required keys of the adversary for the corrupt and honest AA with the assistance of CA. In this process, the adversary's requirements are as follows: the adversary can have the right to obtain the public and private key pairs $\{AA_{pk}, AA_{sk}\}$ of the set of all corrupt authorities $S'_{AA} \subset S_{AA}$, but only the public keys of the remaining authorities $\tilde{S} : (S_{AA} - S'_{AA})$. According to the above requirements, the challenger reasonably calls the system algorithm to complete the generation of the corresponding keys and sends it to the adversary $A$.
- *Query phase 1:* In phase 1, the adversary $A$ makes some queries to challenger $C$ as follows.

  1 Private key query: The adversary can request a user $SK_{u_{id}}$ attribute set $S_1, ..., S_q$ corresponding to some corrupt authorities and obtain his private key from it. However, it is necessary to ensure that there is at least one uncorrupted authority to prevent the adversary from obtaining a sufficient number of keys. Finally, $SK_{u_{id}}$ will be recorded in a set $L_{SK}$, which was created specifically to hold these keys.

  2 Decryption query: The adversary can also perform a decryption query against a given ciphertext CT and output the message $M$ corresponding to the CT. If the ciphertext is incorrectly constructed, $\perp$ is output.

- *Challenge:* After the above process is completed, the adversary $A$ submits $(M_0, M_1, T^*)$ to the challenger, of where $M_0$ and $M_1$ are two messages of equal length, and $T^*$ is an access tree structure other than that specified by the adversary before, and all the sets $S_1, ..., S_q$ in phase 1 do not satisfy this access structure. Now the challenger randomly selects $b \in \{0, 1\}$ and $K \in G_T$, and performs symmetric encryption $E_K(M_b)$ of the message $M_b$ under the access tree structure $T^*$ sent by the adversary, and finally constructs the corresponding $CT^*$. If there exits $SK_i \in L_{SK}$ that can decrypt $CT*$, then the challenge will be terminated. Otherwise, the ciphertext $CT^*$ is given to the adversary.
- *Query phase 2:* Repeating phase 1, the adversary can continue the above private key queries without being able to make decryption queries.
- *Guess:* After the above operations, finally, the adversary guess which message the challenger encrypted, and output his guess $b'$ of $b$, if $b' = b$, then the adversary wins the game.

We define the advantage of the adversary in above game as:

$$Adv^{CCA}_{PPADMA}(A) = |\Pr(b' = b) - 1/2| \quad (5)$$

*Definition 4:* A ABE with security auditing, dynamic multi-authority, and privacy protection scheme is said to be selectively ciphertext policy and chosen-ciphertext attacks (sCP-IND-CCA secure) if for any adversary $A$ within polynomial time, the $Adv^{CCA}_{PPADMA}(A)$ is negligible.

## 5 The proposed scheme

### 5.1 System initialisation

The system initialisation stage is divided into four parts, the algorithm details of each part are as follows.

#### 5.1.1 $CASetup(1^\lambda) \rightarrow ((CA_{sk}, CA_{pk}), (CA_{sig}, CA_{verify}))$

The CA inputs a security parameter $\lambda$ and generates a bilinear mapping $e : G_1 \times G_2 \rightarrow G_T$, where $G_1$ and $G_2$ are two additive groups, $G_T$ is a multiplicative group. Then the public parameters $PP = \{G_1, G_2, G_T, g, h, e(g, h)\}$ are obtained, where $g \in G_1$ and $h \in G_2$ respectively, and $G_1, G_2, G_T$ are with prime order $q$. The CA performs the steps as follows:

Step 1 Choose two random elements $\alpha, \beta$ from $Z^*_q$.

Step 2 Selects two one-way collision-resistance hash functions, called $H_1$ and $H_2$:

$$H_1 : \{0,1\}^* \rightarrow {Z_q}^*$$
$$H_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_\lambda}$$

where $l_\lambda$ is the length of random string.

Step 3 The algorithm generates $(CA_{sk}, CA_{pk})$ as follows:

$$CA_{sk} = \{\alpha, \beta\},$$
$$CA_{pk} = \{PP, e(g,h)^{\alpha\beta}, f = e(g,h)^{\alpha(\beta-1)}, g^\alpha, H_1, H_2\}$$

Step 4 The CA produces a pair of central signature keys and verification keys $(CA_{sig}, CA_{verify})$ by using algorithm like RSA.

After the subsequent $AAEnrol()$ phase is deployed, the CA still needs to initialise some parameter settings in the systems. Please refer to the $AAEnrol()$ phase for details.

#### 5.1.2 $AAEnrol(CA_{sk}, CA_{pk}, AA_{\text{inf}o}) \rightarrow (AA_i, AA_{i,sk}, AA_{i,pk})$

Each authority sends a request to CA to register. The algorithm inputs central authority public and private key

pair $\{CA_{sk}, CA_{pk}\}$, and the information of attribute authority. The CA runs the algorithm and generates a global unique identity for each legitimate authority in the system. As shown in Table 1, let $S_{A_i} = \{A_{i,1}, A_{i,2}, ..., A_{i,n_i}\}$ represents the set of attribute values maintained by $i^{\text{th}}$ attribute authority. Every AA runs the algorithm as follows:

Step 1 $AA_i$ picks a random $c_i \in Z_q$ and computes $x_i, y_i$ as follows:

$$x_i = g^{c_i},\ y_i = h^{c_i}$$

Step 2 Then generates $AA_{i,sk}$ and $AA_{i,pk}$ as:

$$AA_{i,sk} = c_i, AA_{i,pk} = \{x_i, y_i\}$$

Step 3 When the attribute authority is set, CA then sets two default users in this step to be used as the credential information to bind the user and AA together.

1 Set $U = \{0, 1\}$ and randomly some elements $\{v_{u_{id},j}\}_{\forall u_{id} \in U, j \in S_{A_i}}$ and $\{\sigma_i\}_{\forall i \in S_{AA}}$ in $Z_q^*$.

2 Compute

$$\begin{cases} \left\{V_j = \left(\prod_{\forall j \in S_{A_i}} v_{u_{id},j}\right) h\right\} \\ \left\{\bar{v}_{u_{id},j} = \sigma_i \prod_{\forall k \neq u_{id}, k \in U} v_{k,j}^{-1} + v_{u_{id},j} \bmod q\right\}_{\forall j \in S_{A_i}} \end{cases} \tag{6}$$

Finally, the public key of CA is newly changed. The PK contains the information about $\{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j \in S_{AA}}\}$, which is used for subsequent operations. Also, we can view this part as some kind of credential information in the process of system dynamic change. The newly changed PK is as follows:

$$CA_{pk} = \{PP, e(g,h)^{\alpha\beta}, f = e(g,h)^{\alpha(\beta-1)}, g^{\alpha}, H_1, H_2, \{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j \in S_{A_i}}\}\}.$$

### 5.1.3 $UserEnrol(CA_{sk}, CA_{pk}, u_{\text{inf}o}) \rightarrow (u_{id}, cert(u_{id}), u_{sk}, u_{pk})$

Each user who wants to join the system sends a request to the CA. The CA performs the algorithm called $UserEnrol()$ which inputs CA's secret key $CA_{sk}$, public key $CA_{pk}$ and each user information $u_{info}$ and then assigns a unique global identity $u_{id}$ to each user. The algorithm randomly picks a $t_{u_{id}} \in Z_p$, then outputs public and secret key pair of the user $\{u_{pk}, u_{sk}\}$ as follows:

$$u_{sk} = \{t_{u_{id}}\}, u_{pk} = \{h^{t_{u_{id}}}\}$$

At the same time, the algorithm also generates user certificates $cert(u_{id})$ where

$$cert(u_{id}) = \{sign_{\{CA_{sign}\}}(u_{id}, h^{t_{u_{id}}})\}$$

Thus one person can prove that he/she is a legal user in the system by using the certificate $cert(u_{id})$ received from CA.

Then we randomly pick $\{\sigma_i, v_{u_{id},j}\}_{\forall j \in S_{A_i}}$ in $Z_q^*$, set $\{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j \in S_{A_i}}\}$ (we can view this part as some kind of credential information in the process of system dynamic change).

$$\begin{cases} \left\{V_j = \left(\prod_{\forall j \in S_{A_i}} v_{u_{id},j}\right) g\right\} \\ \left\{\bar{v}_{u_{id},j} = \sigma_i \prod_{\forall k \neq u_{id}, k \in U} v_{k,j}^{-1} + v_{u_{id},j} \bmod q\right\}_{\forall j \in S_{A_i}} \end{cases} \tag{7}$$

After a user is registered to the system, then set $U = U \vee \{u_{id}\}$ and update $\{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j \in S_{A_i}}\}$.

## 5.2 Execution

### 5.2.1 $KeyGen(CA_{pk}, AA_{i,sk}, u_{id}, S_{u_{id}}) \rightarrow (SK_{u_{id}})$

Each user registered to the system has its own global identity. They request the decryption key from the AA bound to their own attributes. First, AA uses the CA verification key to verify the user's legal identity, and obtains the user's ID. If the user is found to be illegal, the algorithm aborts. Otherwise, the AA allocates the key that conforms to its attribute value according to the relevant information.

Step 1 Randomly pick some elements $\{h_{u_{id},j},\ \vartheta, \{\rho_i, \sigma_i, t_i, c_i\}_{\forall i \in S_{AA}}, \{v_{u_{id},j}\}_{\forall j \in S_{u_{id}} \bigcap S_{A_i}}, \{r_{i,j}\}_{\forall i \in S_{AA} \forall j \in S_{u_{id}} \bigcap S_{A_i}}\}$ in $Z_q^*$.

Step 2 Compute

$$\begin{cases} \{V_j = v_{u_{id},j} V_j\}_{\forall j \in S_{u_{id}} \bigcap S_{A_i}} \\ \{\bar{v}_{u_{id},j} = \sigma_i \prod_{\forall k \neq u_{id}, k \in U} v_{k,j}^{-1} + v_{u_{id},j} \bmod q\}_{\substack{\forall i \in S_{AA}, \\ \forall j \in S_{u_{id}} \bigcap S_{A_i}}} \\ \{\bar{v}_{k,j} = (\bar{v}_{k,j} - v_{k,j}) v_{u_{id},j}^{-1} + v_{k,j} \bmod q\}_{\substack{\forall k \neq u_{id}, k \in U, \\ \forall j \in S_{u_{id}} \bigcap S_{A_i}}} \end{cases} \tag{8}$$

Step 3 According to the relevant information about user $u_{id}$ and then compute his/her decryption key as follows:

$$\begin{cases} D_{u_{id}} = (\alpha h + \rho_i \sigma_i c_i H_1(u_{id}) h)_{\forall i \in S_{AA}} \\ \{D_{u_{id},j} = v_{u_{id},j}^{-1} (\rho_i + r_{i,j}) \cdot H_1(u_{id})\}_{\forall i \in S_{AA}, \forall j \in S_{u_{id}} \bigcap S_{A_i}} \\ \{D'_{u_{id},j} = r_{i,j} g \cdot H_1(u_{id})\}_{\forall i \in S_{AA}, \forall j \in S_{u_{id}} \bigcap S_{A_i}} \\ \{D''_{u_{id},j} = \sigma_i c_i r_{i,j} g + \rho_i c_i g v_j\}_{\forall i \in S_{AA}, \forall j \in S_{u_{id}} \bigcap S_{A_i}} \end{cases} \tag{9}$$

Step 4 It outputs the secret key

$$SK_{u_{id}, j \in S} = \{D_{u_{id}}, D'_{u_{id}}, D_{u_{id},j}, D'_{u_{id},j}, D''_{u_{id},j}\}_{\forall j \in S_{u_{id}} \bigcap S_{A_i}}$$

### 5.2.2 $Update(u_{id}, j, m'_{u_{id},j}) \rightarrow (SK_{u_{id},j})$

The updating algorithm can solve the issue of the user updating its attribute value and meet the dynamic demand change. With the dynamic changes of users' identities, roles, capabilities and other cases, the attributes of the user will inevitably change. Therefore, users can update

their corresponding attribute values according to the current scenario, such as updating their $j^{\text{th}}$ attribute value to $m'_{u_{id},j}$ to meet the needs from the new environment. The algorithm performs the following operations:

Step 1 Firstly, randomly select some elements, ${\rho_i}', v'_{u_{id},j}$, and $r'_{i,j}$ in $Z_q^*$

Step 2 Next, compute $h'_{u_{id},j} = H_2(m'_{u_{id},j})$

Step 3 Then, give

$$\begin{cases} D_{u_{id}} = (\alpha h + {\rho_i}'\sigma_i c_i H_1(u_{id})h)_{\forall i\in S_{AA},\forall j\in S_{A_i}} \\ \{D_{u_{id},j} = {v'}^{-1}_{u_{id},j}({\rho'}_i + r'_{i,j}h_{u_{id},j}) \cdot H_1(u_{id})\}_{\forall i\in S_{AA},\forall j\in S_{A_i}} \\ \{D_{u_{id},k'} = v^{-1}_{u_{id},k'}({\rho'}_i + r'_{i,k}h_{u_{id},k'}) \cdot H_1(u_{id})\}_{\substack{\forall i\in S_{AA},\\ \forall k'\in S_{A_i}\setminus\{j\}}} \\ \{D'_{u_{id},j} = r'_{i,j}gH_1(u_{id})\}_{\forall i\in S_{AA},\forall j\in S_{A_i}} \\ \{D'_{u_{id},k'} = r'_{i,k'}gH_1(u_{id})\}_{\forall i\in S_{AA},\forall k'\in S_{A_i}\setminus\{j\}} \\ \{D''_{u_{id},j} = h'_{u_{id},j}(r'_{i,j}\sigma_i c_i g + {\rho'}_i c_i g v_j)\}_{\forall i\in S_{AA},\forall j\in S_{A_i}} \\ \{D''_{u_{id},k'} = h'_{u_{id},k'}(r'_{i,k'}\sigma_i c_i g + {\rho'}_i c_i g v_j)\}_{\substack{\forall i\in S_{AA},\\ \forall k'\in S_{A_i}\setminus\{j\}}} \end{cases} \tag{10}$$

to user $u_{id}$.

Step 4 Lastly, update $\{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j\in S_{A_i}}\}$

$$\begin{cases} \left\{V_j = v^{-1}_{u_{id},j}{v'}_{u_{id},j}V_j\right\}_{\forall j\in S_{A_i}} \\ \{\bar{v}_{u_{id},j} = (\bar{v}_{u_{id},j} - v_{k,j}) + v^{-1}_{u_{id},j} \bmod q\}_{\substack{\forall k\neq u_{id},\\ k\in U,\forall j\in S_{A_i}}} \\ \{\bar{v}_{k,j} = (\bar{v}_{k,j} - v_{k,j})v_{u_{id},j}v^{-1}_{u_{id},j} + v_{k,j} \bmod q\}_{\substack{\forall k\in U\setminus\{u_{id}\},\\ \forall j\in S_{A_i}}} \end{cases} \tag{11}$$

### 5.2.3 $Revoke(u_{id})$

After the algorithm is executed, the user identity is no longer valid. In other words, the user cannot legally exist in the system and the user certificate and private key are revoked. In addition, the credentials of the system will be modified and all relevant information about him/her will be deleted. The algorithm update $\{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j\in S_{A_j}}\}$ as follows:

$$\begin{cases} \left\{V_j = v^{-1}_{u_{id},j}V_j\right\}_{\forall j\in S_{A_i}} \\ \{\bar{v}_{k,j} = (\bar{v}_{k,j} - v_{k,j})v_{u_{id},j} + v_{k,j} \bmod q\}_{\substack{\forall k\in U\setminus\{u_{id}\},\\ \forall j\in S_{A_i}}} \end{cases} \tag{12}$$

And then, set $U = U\setminus\{u_{id}\}$ and delete $\{\bar{v}_{u_{id},j}\}_{\forall j\in S_{A_i}}$ in PK.

## 5.3 Encryption and decryption

### 5.3.1 $Encrypt(CA_{pk}, AA_{i,pk}, T, K, M) \to (CT)$

In this scheme, ciphertext generation is carried out under the access structure tree, so the generating process of ciphertext in this paper is divided into two parts: leaf node and internal node.

a *Access tree structure construction Goyal et al. (2006):* let $T$ represent an access structure tree. For a given tree $T$, start from the root node $R$ and select a polynomial $q_x$ for each node(including leaves) of $T$ in a top-down manner. For each node $x$ in the tree, set $d_x = k_x - 1$. The detailed process is described as follows:

- For the root node $R$: randomly chooses an element $s \in Z_q^*$ and sets $q_R(0) = r$, where $k_R$ is the threshold value of the root node $R$, and the process starts from the root node $R$. Then assign a unique index number $x$ for each child of the root node $R$, randomly chooses $d_R$ other points of the polynomial $q_R$ to fully define it.
- For each non-leaf node $N$ other than $R$: randomly pick a polynomial $q_N$ of degree $d_N = k_N - 1$ with $q_N(0) = q_{parent(N)}(index(N))$, where $k_N$ is the threshold value of node $N$. Then assign a unique index number $x$ for each child of the node $N$, randomly chooses $d_N$ other points of the polynomial $q_N$ randomly to fully define it.
- For each leaf node $N_L$: randomly choose a polynomial $q_{N_L}$ of degree 0 with $q_{N_L}(0) = q_{parent(N_L)}(index(N_L))$

b *Ciphertext generation:* randomly select a session key $K \in G_T$ and $K$ is randomly divided into $K_1$ and $K_2$. The $K_2$ is saved by user and $K_1$ is used in ciphertext construction. Then pick a one-way trapdoor function $F$, which is a deterministic algorithm, and its inverse can be calculated when the trapdoor is known. At the same time, we use the way of aggregating the ciphertexts of leaf nodes to reduce the overall ciphertext size to be constant, which is also an improvement over the previous studies.
Then the resulting ciphertext is as follows:

$$\begin{aligned} &CT = \{T, \tilde{C} = e(g,h)^{\alpha\beta r}K_1, C = rg, C' = f^r, \\ &\bar{M} = E_K(M), C_F = F(x_i, K_1 \oplus \bar{M}), \\ &\{C_N = q_N(0)V_{Att(N)}x_i, \\ &C'_N = q_N(0)H_2(val(N))y_i, \\ &C''_N = q_N(0)H_2(val(N))h\} \\ &\{\bar{v}_{u_{id},Att(N)}\}_{\forall u_{id}\in U}\}_{\forall N\in N_L}\} \end{aligned} \tag{13}$$

where $C_N$ is the sum of ciphertexts at the aggregated leaf nodes, $N_L$ is the set of all leaf nodes, $t_N$ and $\sigma_N$ are the leaf node random values corresponding to AA in the key generation phase, and $\vartheta$ is the random value.

### 5.3.2 $Decrypt_{ODCSP}(CT, SK_{u_{id}}) \to (K_1)$

When the user obtains the ciphertext CT from the CSP, because the ciphertext decryption has some exponential operation, our scheme tends to transfer the overhead calculation to the OD-CSP. After the cloud server gets the ciphertext, it calculates the key elements by combining the access policy and the attribute value given by the user with the computing power that is not doped into the system. The

algorithm inputs specific $u_{id}$ ciphertext $CT$ and key $SK$ obtained when users interact with different AA, and outputs the partial session key $K_1$. We specify the decryption procedure as a recursive algorithm.

Before backtracking to calculate the root node, we take a bottom-up approach to calculate it, so we first define a recursive algorithm $DecryptNode(CT, SK, N)$ that takes the ciphertext CT, the private key $SK_{u_{id}}$ of user $u_{id}$, which $SK_{u_{id}}$ is associated with a set $S$ of attributes, and a node $N$ from $T$. $DecryptNode(CT, SK_{u_{id}}, N)$ is define below.

If $N$ is a leaf node: Then we aggregate the target leaf nodes for calculation. If $j \in S_{A_i}$, then:

$$
\begin{aligned}
&DecryptNode(CT, S_{K_{u_{id}}}, N)\\
&= \frac{e(D_{u_{id},j}, \overline{v}_{u_{id},j} \cdot C_N)}{e(D'_{u_{id},j}, C'_N)e(D''_{u_{id},j}, C''_N)}\\
&= \begin{cases} \frac{e(v_{u_{id},j}^{-1}(\rho_N + r_{N,j})H_1(u_{id})h, (\sigma_N v_j^{-1} v_{u_{id},j} + v_{u_{id},j})q_N(0)v_j g^{c_N})}{e(r_{N,j} g H_1(u_{id}), q_N(0) H_2(val(N))h^{c_N})} \cdot \\ \frac{1}{e(\sigma_N c_N r_{N,j} g \cdot H_1(u_{id}) + \rho_N c_N g v_j \cdot H_1(u_{id}), q_N(0)h)} \end{cases}\\
&= \begin{cases} \frac{e(v_{u_{id},j}^{-1}(\rho_N + r_{N,j})H_1(u_{id})h, (\sigma_N v_j^{-1} v_{u_{id},j} + v_{u_{id},j})q_N(0)v_j g^{c_N})}{e(h, g^{c_N})^{r_{N,j} H_1(u_{id}) q_N(0) v_j}} \cdot \\ \frac{1}{e(g\sigma_N r_{N,j} H_1(u_{id}), hq_N(0)c_N)e(g\rho_N v_j H_1(u_{id}), hq_N(0)c_N)} \end{cases}\\
&= \frac{e(h,g)^{H_1(u_{id})(\rho_N + r_{N,j})\sigma_N q_N(0) c_N} e(h,g)^{H_1(u_{id})(\rho_N + r_{N,j}) q_N(0) v_j c_N}}{e(g,h)^{c_N r_{N,j} H_1(u_{id}) q_N(0) v_j} e(g,h)^{\sigma_N r_{N,j} H_1(u_{id}) q_N(0) c_N} e(g,h)^{\rho_N v_j H_1(u_{id}) q_N(0) c_N}}\\
&= \begin{cases} \frac{e(g,h)^{H_1(u_{id})\rho_N \sigma_N q_N(0) c_N} e(g,h)^{H_1(u_{id}) r_{N,j} \sigma_N q_N(0) c_N}}{e(g,h)^{c_N r_{N,j} H_1(u_{id}) q_N(0) v_j} e(g,h)^{\sigma_N r_{N,j} H_1(u_{id}) q_N(0) c_N} e(g,h)^{\rho_N v_j H_1(u_{id}) q_N(0) c_N}} \cdot \\ e(g,h)^{H_1(u_{id})\rho_N q_N(0) v_j c_N} e(g,h)^{H_1(u_{id}) r_{N,j} q_N(0) v_j c_N} \end{cases}\\
&= e(h,g)^{H_1(u_{id})\rho_N \sigma_N q_N(0) c_N}
\end{aligned}
\tag{14}
$$

If $j \notin S_{A_i}$, then we define

$$DecryptNode(CT, SK_{u_{id}}, N) = \perp. \tag{15}$$

If $N$ is an internal node: For all nodes $N_c$ that are children of $N$, it calls $DecryptNode(CT, SK_{u_{id}}, N_c)$ and stores the output as $F_{N_c}$. Let $S_N$ be an arbitrary $k_n$-sized set of child nodes $N_c$ such that $F_{N_c} \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$.

Otherwise, we compute

$$
\begin{aligned}
F_N &= \prod_{N_c \in S_N} F_{N_c}^{\Delta_{n,S'_N}(0)}\\
&= \prod_{N_c \in S_N} \left(\omega^{\rho_N \sigma_N c_N H_1(u_{id}) \cdot q_{N_c}(0)}\right)^{\Delta_{n,S'_N}(0)}\\
&= \prod_{N_c \in S_N} \left(\omega^{\rho_N \sigma_N c_N H_1(u_{id}) \cdot q_{parent(N_c)}(index(N_c))}\right)^{\Delta_{n,S'_N}(0)}\\
&= \prod_{N_c \in S_N} \omega^{\rho_N \sigma_N c_N H_1(u_{id}) \cdot q_N(n) \cdot \Delta_{n,S'_N}(0)}\\
&= \omega^{\rho_N \sigma_N c_N H_1(u_{id}) \cdot q_N(0)}
\end{aligned}
\tag{16}
$$

where $\omega = e(g,h)$, $n = index(N_c)$, $S'_N = \{index(N_c) : N_c \in S_N\}$.

Now we set $A = DecryptNode(CT, SK_{u_{id}}, R)$ when the tree is satisfied by $S$, and then returns

$$
\begin{aligned}
A &= DecryptNode(CT, SK_{u_{id}}, R)\\
&= e(g,h)^{\rho_R \sigma_R c_R H_1(u_{id}) \cdot q_R(0)}\\
&= e(g,h)^{\rho_R \sigma_R c_R H_1(u_{id}) \cdot q_R(0)}
\end{aligned}
\tag{17}
$$

Finally the partial session key $K_1$ can be obtained by computing the formula $A \cdot \tilde{C}/(e(C, D_{u_{id}}) \cdot C')$ where

$$
\begin{aligned}
&\frac{A \cdot \tilde{C}}{e(C, D_{u_{id}}) \cdot C'}\\
&= \frac{e(g,h)^{\rho_R \sigma_R c_R H_1(u_{id}) \cdot r} \cdot e(g,h)^{\alpha\beta r} K_1}{e(rg, \alpha h + \rho_R \sigma_R c_R H_1(u_{id}) h) e(g,h)^{\alpha(\beta-1)r}}\\
&= \frac{e(g,h)^{\rho_R \sigma_R c_R H_1(u_{id}) \cdot r} \cdot e(g,h)^{\alpha\beta r} K_1}{e(g,h)^{\alpha r + \rho_R \sigma_R c_R H_1(u_{id}) r} e(g,h)^{\alpha(\beta-1)r}}\\
&= \frac{e(g,h)^{\rho_R \sigma_R c_R H_1(u_{id}) \cdot r} \cdot e(g,h)^{\alpha\beta r} K_1}{e(g,h)^{\rho_R \sigma_R c_R H_1(u_{id}) r} e(g,h)^{\alpha\beta r}}\\
&= K_1
\end{aligned}
\tag{18}
$$

### 5.4 Audit

#### 5.4.1 $Audit_{AA}(AA_{i,sk}, CT, Decrypt_{ODCSP}(CT, SK_{u_{id}})) \rightarrow (1/0)$

AA runs the algorithm after receiving the result $K_1$ output by outsourced decryption CSP. Then AA calculates the one-way trapdoor function by using its private key to obtains the XOR result of the function value and the partial session key $K_1$. If $F^{-1}(c_i, C_F) \oplus K_1 = \bar{M}$, then it means that the decryption result of the outsourced decryption CSP is correct temporarily, and the values of the partial session key $K_1$ and $\bar{M}$ are consistent and unified, so the preliminary audit is correct and then the algorithm returns 1; Otherwise, it indicates that the outsourced decryption CSP has an error and returns 0. Finally, the AA cannot decrypt a plaintext message even if it retains part of the session key $K_1$.

The rationale for double auditing is as follows: why should AA serve as an auditor when it is not fully trusted? Cannot we just rely on CA for auditing? Consider a scenario where a large number of users need to decrypt simultaneously at the beginning. If all decryption requests were forwarded to CA for processing, it would cause significant overhead and put the CA under a lot of pressure. Therefore, we propose a method where the partial session key, denoted as $K_1$, obtained from ODCSP is first sent to AA for processing. AA's computation can be done in parallel, allowing for the cost of operations to be split between AA and CA. Consequently, the final overhead of CA is reduced to running a symmetric decryption algorithm. The intermediate differences will be elaborated in detail later.

#### 5.4.2 $Audit_{CA}(Audit_{AA}(*) == 1, CA_{pk}, CT, K, M) \rightarrow (1/0)$

After receiving the preliminary audit results from AA, then CA calls the algorithm $Audit_{CA}$. The algorithm will judge the result returned by the $Audit_{AA}$, if it returns 1, the algorithm can be successfully executed; otherwise, it will fall back to the previous step. The algorithm takes the CA's public key, the ciphertext CT, and the session key

$K$ (since the CA is absolutely trusted, the CA can obtain the partial session key $K_2$ of the user. Combined with the partial session key $K_1$ of the ODCSP, the complete session key $K$ is obtained). If $E_K(M) = \bar{M}$, that is, the value of $\bar{M}$ and $M$ are kept unified. Tracing back to the source shows that both the outsourcing decryption CSP and the AA are calculated correctly, then the algorithm returns 1. At the same time, CA returns the correct partial session key $K_1$ transmitted by the outsourcing decryption CSP to DU, and DU recovers the plaintext $M$ by using the symmetric decryption algorithm; Otherwise, if $E_K(M) \neq \bar{M}$, it indicates that there is collusion between the outsourcing decryption CSP and AA, or at least one party has calculation error. Then DU can launch the outsourcing calculation request again and repeat steps 5.6 and 5.7 until the algorithm $Audit_{AA}$ outputs 1, and then call the algorithm $Audit_{CA}$ for audit.

### 5.5 Decryption by user

#### 5.5.1 $Decrypt_{DU}(Audit_{CA}(*) == 1, cert(u_{id}), CT, K_1, K_2) \rightarrow (M/\perp)$

The algorithm will judge the result returned by the $Audit_{CA}$, when the algorithm $Audit_{CA}$ outputs 1, then user $u_{id}$ submits his partial session key $K_2$ and own identity certificate. After the verification is passed, he will receive the correct partial session key $K_1$ jointly audited by AA and CA, then recover the complete session key according to the two partial session keys, and finally the plaintext $M$ can be recovered by using the symmetric decryption algorithm, that is $D_K(\bar{M}) = M$.

## 6 Security analysis

In this section, we will focus on proving that the scheme in our paper is selectively secure under the previously defined security model.

*Theorem 1:* The proposed PPADMA-ABE scheme is secure against chosen-ciphertext attack(CCA) in selective model under the DBDH assumption.

*Proof:* Assuming that there exists an adversary that can break our scheme in probabilistic polynomial time, then we can construct an algorithm to break the DBDH assumption by taking advantages of the adversary's capability. Suppose we have an adversary $A$, which has a non-negligible advantage $\epsilon = AdvA$ in the selective security game against our scheme. Here, we assume that the challenger is given relevant parameters. If the challenger wants to break the DBDH assumption, he needs at least $\frac{1}{2} + \varepsilon$ probability to determine whether $Z = e(g,h)^{abc}$ or not. We now build a simulator $B_e$ that plays DBDH problem. The details as follows:

- *Init:* Given a DBDH instance $(G_1, G_2, G_T, q, g, h, e, ag, bg, cg, ah, bh, ch, Z)$, the challenger follows step 3 of the $AASetup()$ algorithm to generate two users and $\{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j \in S_{A_i}}\}$. Meanwhile, the challenger publishes $PK = \{G_1, G_2, G_T, e, H, g, h, f = e(\alpha g, \beta h - h), e(\alpha g, \beta h), \{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j \in S_{A_i}}\}\}$ and set the master key $MK = \{\alpha, \beta\}$. The simulator $B_e$ inputs a DBDH challenge and other parameters, and the adversary selects a challenge access policy $T^*$ and sends it to the simulator. After that, the simulator creates a list $L_{SK}$ for storing the subsequent values, and simulates the oracles in Phase 1.
- *Setup:* The challenger runs $CASetup()$ and $AASetup()$. For all corrupted AAs $(A_k \in C_A)$: The simulator $B_e$ picks $c_k \leftarrow Z_q$ and sets $X_k = g^{c_k}$, $Y_k = h^{c_k}$. Therefore, the public-secret key pairs for $A_k \in C_A$ is given as $\{(c_k), (X_k = g^{c_k}, Y_k = h^{c_k})\}$. Then the simulator $B_e$ provides the public-secret key pairs of the corrupted AAs to the adversary. Now the adversary can get the $SK_{u_{id},k}(D_{u_{id}}, D_{u_{id},k}, D'_{u_{id},k}, D''_{u_{id},k})$ alone for user $u_{id}$. The simulator only sends public key to the adversary for the remaining AAs that are not corrupted.
- *Phase 1:* The simulator $B_e$ using the $KeyGen()$ algorithm to get the private key $SK_{u_{id}}$ according to $S_i$ for each AAs. When the adversary need to ask for the private key of the $i^{\text{th}}$ authority corresponding to a user $u_{id}$ that he wants to query, then the $B_e$ return $SK_{u_{id},i}$ and store $SK_{u_{id},i}$ in $L_{SK}$. When the adversary makes the decryption query, he will send the ciphertext to the simulator in advance, and simulator $B_e$ will judge whether it's a correctly constructed ciphertext, and if not, the output will be $\perp$. Otherwise, the simulator $B_e$ first checks whether there exists a private key $SK_{u_{id},i}$ in $L_{SK}$ that can decrypt CT. If so, $B_e$ decrypts CT by using $SK_{u_{id},i}$ and then returns the final result to $A$. If not, then the simulator $B_e$ will create a pseudo-user $w$ and take user $w$'s private key to decrypt the ciphertext CT. Details are as follows.

  1 Randomly select the following elements which are

  $$\begin{cases} \{\rho_i, \sigma_i\}_{\exists i \in S_{AA}} \in Z_q^* \\ \{v_{w,j}, r_{i,j}\}_{\exists i \in S_{AA}, j \in S_{u_{id}} \bigcap S_{A_i}} \in Z_q^* \end{cases}$$

  2 Compute

  $$\begin{cases} \{C_N = v_{w,att(N)} C_N\}_{\forall N \in N_L} \\ \left\{ \bar{v}_{w,j} = \sigma_w \prod_{\forall k \in U} v_{k,j}^{-1} + v_{w,j} \bmod q \right\}_{j \in S_{u_{id}} \bigcap S_{A_i}} \end{cases} \tag{19}$$

  where $C_N$ is retrieved from CT and refreshed the credential information for pseudo-users w in this step.

3 Calculate the private key $SK_{u_w}$ of pseudo user $w$ as follows:

$$\begin{cases} D_{u_w} = \alpha h + \rho_i \sigma_i c_i H_1(u_w) h \\ \{D_{u_w,j} = h v_{w,j}^{-1}(\rho_i + r_{i,j}) \cdot H_1(u_w)\}_{\forall j \in S_{u_{id}} \bigcap S_{A_i}} \\ \{D'_{u_w,j} = r_{i,j} g \cdot H_2(h^{t_{u_w}})\}_{\forall j \in S_{u_{id}} \bigcap S_{A_i}} \\ \{D''_{u_{id},j} = \sigma_i c_i r_{i,j} g + \rho_i c_i g v_j\}_{\forall j \in S_{u_{id}} \bigcap S_{A_i}} \end{cases} \quad (20)$$

4 Take user $w$'s private key $SK_{u_w}$ to decrypt the CT and return the final result to adversary.

**Algorithm 1** The $challenge\_CT(T, N, C_0)$ algorithm

**for** each $N \in T$ **do**
  initialise a tree $T^*$ with only a leaf (the root);
  $T = T \cup T^*$;
**end for**
**if** ($N$ is a leaf node) **then**
  Set $j = att(N)$ and $m = val(N)$
  Compute $C_N = v_{att(N)} C0_{\forall N \in N_L}$
  Compute $C'_N = q_N(0) H_2(val(N))$
  Compute $C''_N = H_2(val(N)) v_{att(N)}$
  Store $(\{C_N, C'_N, C''_N\}_{N \in N_L})$ in $L_C$
**else**
  Randomly select $k_N - 1$ elements $d_i \in Z_q^*$
  **for** each child $N_C$ of the node $N$ **do**
    Set $\eta = index(N_C)$;
    **if** ($k_N - 1 > 0$) **then**
      Compute $C'_0 = C_0 + \sum_{i=1}^{k_N - 1} d_i \eta Q$
    **else**
      $C'_0 = C_0$
    **end if**
    Call $challenge\_CT(T, N_C, C'_0)$
  **end for**
**end if**

- *Challenge:* When all the preset work is ready, the adversary will send two messages $M_0$ and $M_1$ to the simulator. After receiving the $(M_0, M_1, T^*)$, the simulator $B_e$ creates a set list $L_C$ and makes $R$ become the root node of the access tree $T^*$ sent by the adversary. Then the simulator defines a recursive algorithm $challenge\_CT(T^*, R, \alpha Q)$ in Algorithm 1 for recursively computing the ciphertext value at the root node $R$ from the leaf nodes. And it is also used to store the ciphertext value of the challenged node $\{C_N, C'_N, C''_N\}_{N \in N_L}$ in the list $L_C$. Therefore, the final ciphertext node $\hat{C_N}$ value is obtained by the above nodes aggregation. The $B_e$ randomly selects a $b \in \{0, 1\}$ and $K' \in G_T$, and let the message chosen by the challenger be denoted as $M_b$ where $b = 0$ or 1 and calculates the ciphertext
  $CT^* = \{T^*, \tilde{C} = ZK_1, C = \gamma g, C' = \frac{Z}{e(\alpha g, \gamma h)}$,
  $\bar{M_b} = E_{K'}(M_b), C_F = F(x_i, K_1 \oplus \bar{M_b})$,
  $\hat{C_N} = \{C_N, C'_N, C''_N, \{\bar{v}_{u_{id}, Att(N)}\}_{\forall u_{id} \in U}\}_{N \in N_L}\}$,
  where the encrypted value of the node is retrieved from the encrypted list $L_C$. In addition, we emphasise that $CT^*$ is a valid encrypted information of the message $M_b$ if $Z = e(g, h)^{abc}$; otherwise, the $CT^*$ is just random value. If there exists $SK_{u_{id},k} \in L_{SK}$ that can decrypt the ciphertext, the challenge will be aborted; otherwise, the $B_e$ returns the $CT^*$ to A.
- *Phase 2:* The adversary continues to query as in phase 1, but restricts it to not be able to make decryption queries.
- *Guess:* Finally, the adversary $A$ gets his guess $b'$ for $b$. If $b'$=$b$, the $B_e$ outputs 1 and guess that $Z = e(g, h)^{abc}$(means it is a valid BDH-tuple). Otherwise, the adversary is wrong and $B_e$ outputs 0, which denotes the $Z$ is a random element in $G_T$. Thus it can be seen that if an adversary breaks our scheme with non-negligible advantage at least $\varepsilon$ in polynomial time, which means that a challenger has a non-negligible $\varepsilon'$ to break the DBDH assumption where $\varepsilon' \geq \varepsilon - \delta$ and $\delta$ is a negligible advantage. If $Z = e(g, h)^{abc}$, then we can get $\Pr[B_e(g, h, ag, bg, cg, ah, bh, ch, e(g, h)^{abc}) = 1] = \Pr[b' = b]$ where $\left|\Pr[b' = b] - \frac{1}{2}\right| \geq \varepsilon$. Otherwise, the $Z$ is a random element in $G_T$, and we have $\Pr[B_e(g, h, ag, bg, cg, ah, bh, ch, Z) = 1] = \Pr[b' = b]$ with $\left|\Pr[b' = b] - \frac{1}{2}\right| \leq \delta$ where $\delta$ is the advantage of breaking the semantic security of $E_K$. Finally, we have

$$\begin{aligned} &|\Pr[B_e(g, h, ag, bg, cg, ah, bh, ch, e(g, h)^{abc}) = 1] \\ &- \Pr[B_e(g, h, ag, bg, cg, ah, bh, ch, Z) = 1]| \\ &\geq \left|\left(\frac{1}{2} \pm \varepsilon\right) - \left(\frac{1}{2} \pm \delta\right)\right| \\ &\geq \varepsilon - \delta \end{aligned} \quad (21)$$

*Theorem 2 (collusion resistance):* The PPADMA-ABE scheme is collusion-resistant against colluding users or attribute authorities.

*Proof:* For colluding users: In terms of user collusion, the proposed scheme utilises the unique global identity of all users to prevent collusion attacks. In the decryption process, the user needs to calculate $SK_{u_{id}}$ and submit it to the outsourced decryption cloud server, then get the partial session key $K_1$ by computing $e(g, h)^{\rho_R \sigma_R c_R H_1(u_{id}) \cdot q_R(0)}$, where $u_{id}$ is the unique global identity and $q_R(0)$ is the value of the root node of the current access control tree $T$. To achieve this, the collusion user needs to calculate the exponential value on the bilinear pair, which is bound to the user's GID. Hence, if two or more users having different GID try to collude the term $e(g, h)^{\rho_R \sigma_R c_R H_1(u_{id}) \cdot q_R(0)}$ would not cancel out and it is impossible for two users to generate decryption credential for third user.

Without loss of generality, suppose that our scheme involves only three attribute authorities, denoted by $A_1$, $A_2$, and $A_3$, and that there are two users, denoted by $u_1$ and $u_2$. Furthermore, suppose that the data owner (DO) encrypts the content message using the policy $< a_{1,1} \cap a_{2,1} \cap a_{3,1} >$, where for simplicity, we assume that the policy takes a conjunction form. Here, $a_{i,j}$ represents the $j^{\text{th}}$ attribute managed by the $i^{\text{th}}$ attribute authority. It is known that user $u_1$ possesses attribute $a_{1,1}$, while user $u_2$ possesses attributes $a_{2,1}, a_{3,1}$. It is not difficult to observe that neither $u_1$ nor $u_2$ alone can decrypt the content as they do not satisfy the access policy. However, if $u_1$ and $u_2$ collude, they possess all the attributes required to satisfy the access policy, and each of them generates a private key accordingly as follows:

$$SK_{u_1} = \begin{cases} D_{u_1} = \alpha h + \rho_1 \sigma_1 c_1 H_1(u_1) h \\ D_{u_1,1} = v_{u_1,1}^{-1}(\rho_1 + r_{1,1}) \cdot H_1(u_1) \\ {D'}_{u_1,1} = r_{1,1} g \cdot H_1(u_1) \\ {D''}_{u_1,1} = \sigma_1 c_1 r_{1,1} g + \rho_1 c_1 g v_1 \end{cases} \tag{22}$$

$$SK_{u_2} = \begin{cases} D_{u_2} = \alpha h + \rho_2 \sigma_2 c_2 H_1(u_{id}) h \\ \{D_{u_2,j} = v_{u_2,j}^{-1}(\rho_2 + r_{2,j}) \cdot H_1(u_2)\}_{\forall j \in \{2,3\}} \\ \{{D'}_{u_2,j} = r_{2,j} g \cdot H_1(u_2)\}_{\forall j \in \{2,3\}} \\ \{{D''}_{u_2,j} = \sigma_2 c_2 r_{2,j} g + \rho_2 c_2 g v_j\}_{\forall j \in \{2,3\}} \end{cases} \tag{23}$$

Next, they combine their respective private keys and separately invoke the recursive algorithm $DecryptNode()$ on the access tree structure, with the modified key parameter of the combined private keys of the colluding users $u_1$ and $u_2$. The calculation process is as follows.

$$\begin{aligned} & DecryptNode(CT, SK_{u_1} \cup SK_{u_2}, N) \\ &= \frac{e(D_{u_1,1}, \bar{v}_{u_1,1} \cdot C_N)}{e({D'}_{u_2,2}, {C'}_N) e({D''}_{u_2,3}, {C''}_N)} \\ &= \frac{e(v_{u_1,1}^{-1}(\rho_N + r_{N,1}) \cdot H_1(u_1), (\sigma_N v_1^{-1} v_{u_1,1} + v_{u_1,1}) q_N(0) v_1 h g^{c_N})}{e(r_{N,2} g H_1(u_2), q_N(0) H_2(val(N)) h^{c_N})} \\ &\quad \cdot \frac{1}{e(\sigma_N c_N r_{N,3} g \cdot H_1(u_2) + \rho_N c_N g v_3 \cdot H_1(u_2), q_N(0) h)} \\ &= \frac{e(\rho_N h \cdot H_1(u_1), \sigma_N q_N(0) g^{c_N}) e(r_{N,1} h \cdot H_1(u_1), q_N(0) \sigma_N g^{c_N})}{e(r_{N,2} h \cdot H_1(u_2), q_N(0) v_2 g^{c_N}) e(r_{N,3} h \cdot H_1(u_2), q_N(0) \sigma_N g^{c_N})} \\ &\quad \cdot \frac{e(\rho_N h \cdot H_1(u_1), q_N(0) v_3 g^{c_N}) e(r_{N,3} h \cdot H_1(u_1), q_N(0) v_1 g^{c_N})}{e(\rho_N h \cdot H_1(u_2), q_N(0) v_3 g^{c_N})} \\ &= \frac{e(r_{N,1} h \cdot H_1(u_1), q_N(0) v_1 g^{c_N}) e(r_{N,1} h \cdot H_1(u_1), q_N(0) \sigma_N g^{c_N})}{e(r_{N,2} h \cdot H_1(u_2), q_N(0) v_3 g^{c_N}) e(r_{N,3} h \cdot H_1(u_2), q_N(0) \sigma_N g^{c_N})} \\ &\quad \cdot \frac{e(\rho_N h \cdot H_1(u_1), q_N(0) g^{c_N})}{e(\rho_N h \cdot H_1(u_2), q_N(0) g^{c_N})} \cdot e(\rho_N h \cdot H(u_1), \sigma_N q_N(0) g^{c_N}) \end{aligned} \tag{24}$$

The above equation structure reveals that, because the attributes held by $u_1$ and $u_2$ cannot be directly merged for computation, even with colluding private keys, the AA index numbers associated with their respective attributes will affect the calculation of the secret value of the root node subset of the access tree structure. When performing recursive calculations, the exponent values of bilinear pairings cannot be cancelled out (e.g., $r_{N,j}$), and the hash value of the user's GID is also difficult to eliminate, which affects the acquisition of the secret value at the root node of the access tree, and ultimately prevents the acquisition of partial session key $K_1$. This explains why malicious users cannot collude successfully. Therefore, the collusion cannot be realised and our scheme can resist against malicious user collusion.

For colluding AAs: on the one hand, they would need to forge user identities to obtain the hash value of user IDs during the key generation phase. On the other hand, during the decryption phase, the authority would need to obtain two parts of the session key, namely $K1$ and $K2$. Even if an authority could obtain $K_1$ by forging or other means, it would still be challenging to guarantee that it would be recognised during the key extraction agreement with the data owner. Therefore, obtaining the complete session key is difficult, and the collusion is likely to fail.

Furthermore, the scheme involves two audit operations, and the results of AA's initial audit must undergo a second audit by CA. As outlined in Subsection 3.1.2, the main purpose of CA's audit is to ensure that AA's operations are correct and to compare the content from AA audit outsourcing cloud decryption service to detect any collusion. Thus, if malicious AAs collude, CA would detect it during the second audit. As a completely trusted entity in the system, CA is a fundamental guarantee that collusion between malicious AAs cannot succeed.

*Theorem 3 (Backward and forward secrecy):* The proposed PPADMA-ABE scheme promises the backward and forward secrecy against the users.

*Proof:* When a user leaves the system, their credentials are revoked and the code of the revoked attribute is updated. If a new user with a revoked attribute joins the system, their key is bound to the latest credential record of the current system, and the previously published ciphertext is updated with the latest attribute code and credential. This allows the newly added user to decrypt the previously published ciphertext if they satisfy the access policy. As a result, this approach ensures forward security.

However, if a user is revoked, they cannot decrypt the associated ciphertext in the current system even if they still have a decryption key that satisfies the access policy. Additionally, they cannot decrypt updated ciphertext without updating their key. While the updated key is still associated with the user's global identity, a revoked user cannot use the updated keys of other non-revoked users to update their own keys. Therefore, if ciphertext data is downloaded, the user cannot decrypt it as usual. This scheme ensures backward security.

**Table 2** Comparison of security and functionality among different schemes

| *Schemes* | *CA* | *MA* | *OD* | *AU* | *DN* | *CR* | *PP* | *CCS* | *GID* | *MC* | *Hardness* | *Model* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Chase and Chow (2009) | × | √ | × | × | × | √ | √ | × | Privacy | CPA | DBDH | ROM |
| Premkamal et al. (2020b) | √ | × | √ | × | √ | √ | √ | √ | Privacy | CPA | DBDH | ROM |
| Han et al. (2015) | × | √ | × | × | × | × | √ | × | Public | CPA | q-PBDHE q-SDH | ROM |
| Deng (2014) | √ | × | × | × | √ | × | × | × | Public | CPA | q-BDHE | ROM |
| Rahulamathavan et al. (2016) | × | √ | × | × | × | √ | √ | × | Privacy | CPA | DBDH | ROM |
| Fan et al. (2014) | √ | × | × | × | √ | × | √ | × | Privacy | CCA | DBDH | Standard |
| Ling et al. (2021) | √ | √ | × | × | × | √ | √ | × | Public | CPA | q-PBDHE | ROM |
| Ning et al. (2017) | √ | × | √ | × | × | √ | √ | × | Private | CPA | q-BDHI | ROM |
| Our scheme | √ | √ | √ | √ | √ | √ | √ | √ | Privacy | CCA | DBDH | Standard |

Notes: Abbreviated symbol description: MA – multi-authority, OD – outsourced decryption, AU – auditability, DN – dynamicity, CR – collusion resistance, PP – privacy protection, CCS – constant ciphertext size, GID – global identity, MC – message confidentiality.

## 7 Performance comparison

### 7.1 Theoretical analysis

#### 7.1.1 Feature and security comparisons

The performance comparison among the proposed scheme and some existing schemes is shown in Table 2. Clearly, some challenging features, such as traceability, policy updating and data access restrictions, have been addressed in schemes in Premkamal et al. (2020a), Ling et al. (2021) and Lian et al. (2019). However, our proposed scheme also addresses the problem of achieving a good balance between dynamicity and multi-authorities, which enhances the system's practicality and flexibility. Furthermore, we incorporate an auditing feature to ensure the correctness of the decrypted ciphertext by the ODCSP, which helps to improve the system's security. Our research also provides resistance against collusion attacks, thus enhancing user privacy and ensuring the confidentiality of the system.

**Table 3** Notations used in performance evaluation

| *Notation* | *Meaning* |
|---|---|
| $t_p$ | The time for single pairing operation |
| $t_e$ | The time for single exponentiation operation |
| $n_u$ | The number of a user's attributes |
| $n_a$ | The number of authorities |
| $n_\tau$ | The number of attributes in the access tree |
| $n_c$ | The number of attributes associated to a ciphertext |
| $n_{nl}$ | The number of non-leaf nodes in the access tree |
| $\|U\|$ | The number of universal attributes |
| $\|G_*\|$ | The number of the elements in $G_*$ |
| $l$ | The number of rows of the matrix in LSSS scheme |

We can see that the scheme in Deng (2014) has some dynamic features, but it does not realise the protection of user privacy, and does not discuss the collusion situation reasonably. Although, Fan et al. (2014) also has a certain dynamic policy updating, it does not integrate multi-authority attributes, and the practicability of the system needs to be improved. The scheme in Ling et al. (2021) can realise the collusion resistance and protect the privacy, but it does not provide the auditing functionality. And all the schemes suffer from efficiency issues.

#### 7.1.2 Computational analysis

Here, we present a comparison of the computational costs of the proposed scheme with other existing schemes. We define the relevant symbols used in the comparison process in Table 3. Our computational cost analysis includes the pairing and exponentiation operations required for key generation, encryption (user), decryption (user), and outsourced decryption, as detailed in Table 4. The results presented in Table 4 show that our scheme has lower key cost compared to other existing schemes, and it requires fewer exponentiation operations than Fan et al. (2014). Moreover, our DU does not require any exponentiation or pairing operations during the decryption process, thereby significantly reducing the decryption cost for clients. Furthermore, our client-side decryption overhead is much lower than schemes with outsourced decryption, such as Premkamal et al. (2020a) and Ning et al. (2017). In our scheme, the decryption overhead of DU is outsourced to OD-CSP, and the computational cost of this part is $(3n_\tau + 4)t_p + n_{nl}t_e$. Thus, the total decryption cost can be calculated by combining the two results. Compared to other outsourced schemes in Premkamal et al. (2020a) and Ning et al. (2017), our PPADMA-ABE scheme requires less computation time.

The key generation overhead represents the cost required by all attribute authorities to generate decryption keys for users. Encryption (decryption) overhead refers to the combination of all exponentiation and pairing operations used during the encryption (decryption) of messages using access policies. Hash and group operation overheads are negligible and are therefore ignored. From Table 4, we can observe that the computational overhead of all ABE schemes increases linearly with the number of members and attributes.

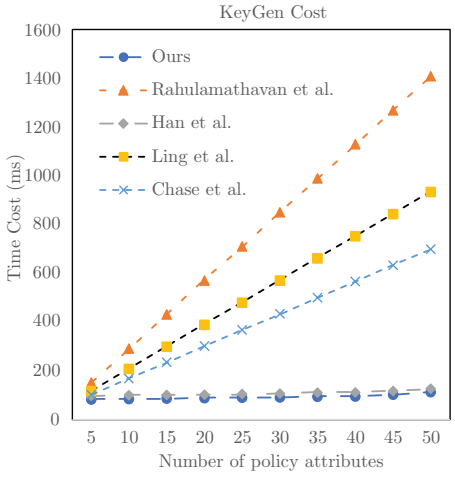
**Table 4** Theoretical comparison of computational cost among different schemes

| *Schemes* | *Key generation* | *Encryption* | *Decryption (user)* | *Outsourced decryption* |
|---|---|---|---|---|
| Chase and Chow (2009) | $({n_a}^2+1)t_e$ | $(n_\tau+2)t_e$ | $(n_a+3)t_p+t_e$ | - |
| Han et al. (2015) | $10n_at_e$ | $(n_a+3)t_e+n_ct_p$ | $(4n_u+2l)t_p$ | - |
| Rahulamathavan et al. (2016) | $5n_ct_e$ | $2n_ct_e+n_ct_p$ | $(n_c+n_a+1)t_p$ | - |
| Ling et al. (2021) | $2n_ut_e$ | $(n_c+1)t_p+3n_ct_e$ | $5n_c(t_p+t_e)$ | - |
| Premkamal et al. (2020a) | $(n_u+3)t_e$ | $(n_\tau+2)t_e$ | $t_e$ | $(n_u+2)t_p+n_{nl}t_e$ |
| Deng (2014) | $(3+n_u)t_e+2t_H$ | $(n_c+1)t_e+t_p$ | $(2n_u+1)t_p$ | - |
| Fan et al. (2014) | 1 | $(2n_c+1)t_e+t_p$ | $(n_u+1)t_p+n_{nl}t_e$ | - |
| Ning et al. (2017) | $(n_u+5)t_e$ | $(5n_\tau+2)t_e$ | $t_e$ | $(3n_\tau+3)t_p+lt_e$ |
| Our scheme | 1 | $(n_c+1)t_e+t_p$ | 1 | $(3n_\tau+4)t_p+n_{nl}t_e$ |

**Table 5** Theoretical comparison of storage size among different schemes

| *Schemes* | *Size of public parameters* | *Size of private key* | *Size of ciphertext* |
|---|---|---|---|
| Chase and Chow (2009) | $\lvert G_1\rvert+\lvert G_2\rvert+\lvert G_T\rvert$ | $(n_u+1)\lvert G\rvert$ | $(n_c+2)\lvert G\rvert$ |
| Premkamal et al. (2020a) | $\lvert G\rvert+\lvert G_t\rvert$ | $(n_u+3)\lvert G\rvert$ | $(n_c+1)\lvert G\rvert+\lvert G_t\rvert$ |
| Han et al. (2015) | $2\lvert G\rvert+\lvert G_T\rvert$ | $6n_u\lvert G\rvert+\lvert G_T\rvert$ | $(2l+3)\lvert G\rvert+\lvert G_T\rvert$ |
| Deng (2014) | $2\lvert G\rvert$ | $(n_u+2)\lvert G\rvert$ | $(l^2+2)\lvert G\rvert$ |
| Rahulamathavan et al. (2016) | $2\lvert G_1\rvert+\lvert G_2\rvert$ | $3n_u\lvert G_1\rvert$ | $(2n_c+1)\lvert G_1\rvert+\lvert G_2\rvert$ |
| Fan et al. (2014) | $\lvert G_0\rvert+\lvert G_1\rvert+\lvert G_T\rvert$ | $3n_u\lvert G_0\rvert+\lvert G_1\rvert$ | $2\lvert G_0\rvert+(3n_c+1)\lvert G_T\rvert$ |
| Ling et al. (2021) | $3\lvert G\rvert+\lvert G_T\rvert$ | $(4n_u+1)\lvert G\rvert$ | $(5l+1)\lvert G\rvert+\lvert G_T\rvert$ |
| Ning et al. (2017) | $2\lvert G_T\rvert$ | $(2n_u+4)\lvert G_T\rvert$ | $(3l+1)\lvert G_T\rvert$ |
| Our scheme | $\lvert G_1\rvert+\lvert G_2\rvert+\lvert G_T\rvert$ | $3\lvert G_1\rvert+n_u\lvert G_2\rvert$ | $\lvert G_*\rvert$ |

**Figure 2** Computational costs comparison among different schemes (see online version for colours)




Moreover, let us recall the discussion in Subsection 4.1 concerning the initial audit by AAs and secondary audit by the CA. Consider a scenario where $n_m$ messages need to be decrypted simultaneously, implying that there are $n_m$ values sent by the cloud server requiring auditing. If we assume that only the CA is responsible for auditing, then the auditing cost of the CA is to compute $n_m$ XOR values of the trapdoor functions and partial decryption key values simultaneously and to perform $n_m$ symmetric decryption algorithms. We denote the cost involved in the CA as $O(n_m)$, where big O notation is used to express the operation cost of auditing. However, if the audit process is split into two steps, and the AA takes the role of the initial auditor, then the XOR overhead originally calculated by the CA is shifted to the AA, and the cost is evenly amortised to each AA, which becomes $O(n_m)/n_a$. Now, the CA only needs to be responsible for the auditing of the results checked by the AA, which is performing only the symmetric decryption algorithm. Therefore, the audit overhead of the CA is $O(1)$, which is a significant improvement. In total, the amortised cost of auditing is $O(n_m)/n_a$, and thus, the overall system efficiency is improved. Hence, this explains why two audits are required from a data perspective, and it provides a qualitative answer to the reasons for requiring two audits.

### 7.1.3 *Storage overhead*

In this section, we mainly consider the comparison of storage costs for public parameters, private key sizes, and ciphertext sizes. The storage sizes associated with past ABE schemes have been shown in detail in Table 5, from which we can observe that the proposed scheme is consistent with the storage overhead used by Fan et al. (2014) and Jarecki and Liu (2009) to store public parameters. Moreover, our work achieves a slightly lower private key size than other schemes while ensuring accurate and secure decryption. One of the factors that affects the storage cost of scheme is the size of the group elements. Furthermore, as shown in Table 5, the cost increases even further if an attribute association coefficient is incorporated prior to this value. Generally, the value of a variable with an attribute coefficient is higher than that of a variable with a constant coefficient, which theoretically leads to a greater overhead for such a scheme. Hence, this information can be used to compare the differences between schemes more intuitively. In order to ensure the high efficiency and security of the

system, how to strike a balance between storage cost and security efficiency also needs to be further balanced and considered.

### 7.2 Experimental analysis

In this section, we conducted simulation experiments to demonstrate the performance differences between the proposed scheme and other schemes. All tests were carried out on a computer running Ubuntu 18.04 with an Intel(R) Core i5-7200U quad CPU, 3.60 GHz, and 8 GB RAM. The simulation experiments were implemented using Python 3.5.2 programming software, with Charm-Crypto library and Pairing-Based Cryptography library as the main function libraries. To standardise the experimental comparison scale and make the comparison results more objective, we selected four ABE schemes with multiple authority settings in the above scheme for comparison. We assumed that the number of users and attribute authority remained constant at 10 and 20, respectively, and gradually increased the number of attributes in the policy from 5 to 50. In the simulation experiment, all simulation results were averaged after being tested 50 times. The unit of storage size and computational consumption in the comparison results are kilobytes (Kb) and milliseconds (ms), respectively.

Figures 2 and 3 display the computational overhead comparison based on the simulation experiment conducted in this paper. Specifically, Figure 2 illustrates the key generation overhead, whereas Figure 3 presents encryption and decryption cost. In the decryption phase of our scheme, since the user overhead is at a constant level, we quantified the decryption overhead into system decryption (outsourcing decryption and user decryption). It is noteworthy that the decryption cost of this scheme is considerably high, as evident from Figure 3(b), which is why we chose to outsource decryption. The storage overhead comparison results are presented in Figure 4, where Figure 4(a) displays the key storage size, and Figure 4(b) shows the ciphertext storage size. Furthermore, the experimental results indicate that the calculation cost of all ABE schemes increases linearly with the growth of the number of attributes in the policy, which is consistent with the theoretical analysis mentioned above.

**Figure 3** Computational costs comparison among different schemes (see online version for colours)

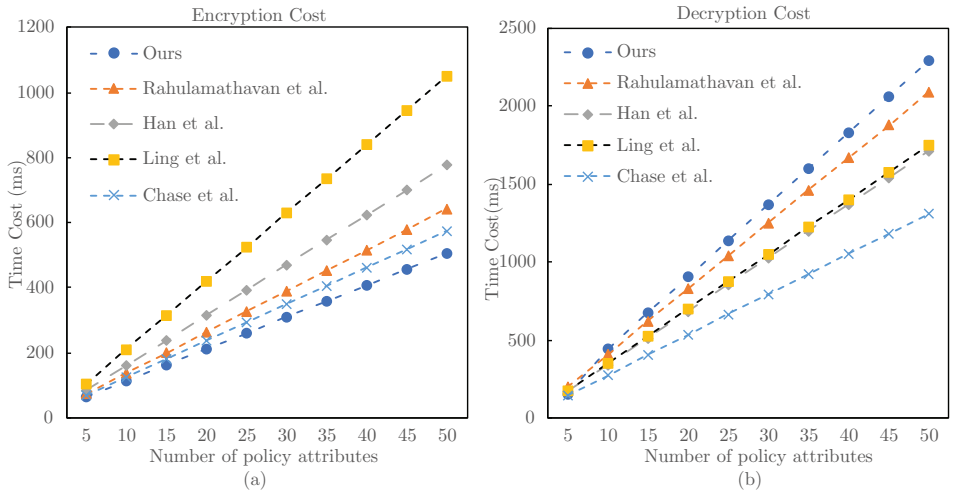
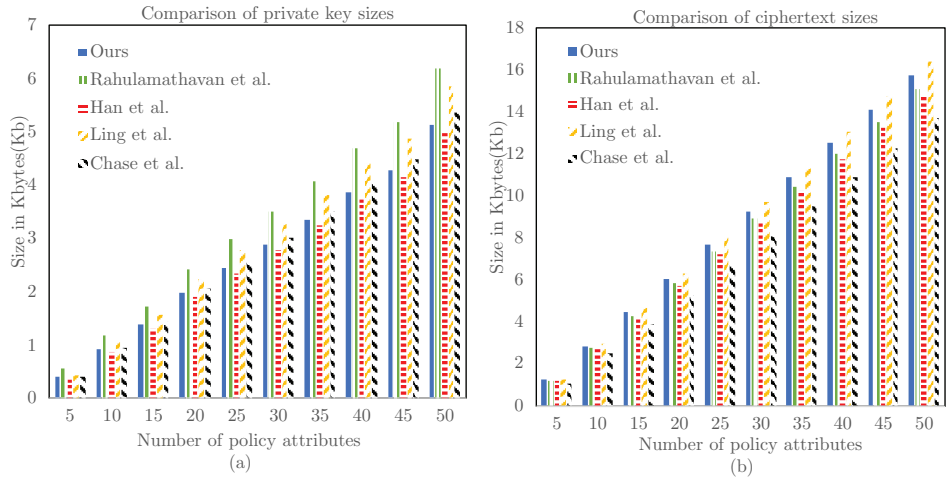



**Figure 4** Storage size comparison among different schemes (see online version for colours)



## 8 Conclusions

Our proposed scheme combines multi-authority and dynamicity based on ABE to optimise the performance of the system and enhance the practicality and flexibility of ABE. Decryption outsourcing is used to reduce system overhead, and the outsourcing results are audited twice without introducing a third party to ensure information integrity, correctness, and system security. The nonlinear coupling of ciphertext in the scheme also makes the system resistant to collusive attacks, preventing malicious authorities and users from combining keys to obtain access rights, which is also proved in our scheme. Moreover, we prove that the PPADMA-ABE scheme can achieve CCA security without the random oracle, and it reduces the decryption cost on the client side to a constant level, demonstrating strong practical scalability. Finally, we also carried out the simulation experiment of the scheme, and the results show that it has a good performance. Thus, our scheme enhances user privacy while resisting collusion from malicious users or authorities, with forward and backward security being realised. These advantages make ABE more efficient and flexible for real-world applications.

## Acknowledgements

This work has been partly supported by self-determined research funds of CCNU from the colleges' basic research and operation of MOE under Grand No. CCNU22JC001.

## References

Armknecht, F., Bohli, J.M., Karame, G.O., Liu, Z. and Reuter, C.A. (2014) 'Outsourced proofs of retrievability', *Computer and Communications Security*, ACM, DOI: 10.1145/2660267.2660310.

Beimel, A. (1996) *Secure Schemes for Secret Sharing and Key Distribution*, PhD thesis, Israel Institute of Technology Technion.

Bethencourt, J., Sahai, A. and Waters, B. (2007) 'Ciphertext-policy attribute-based encryption', *IEEE Symposium on Security & Privacy (SP '07)*, Berkeley, France, May.

Boneh, D. (2001) 'Identity-based encryption from the Weil pairing', *Advances in Crytology, Crypto 2001*.

Boneh, D. and Shoup, V. (2020) *A Graduate Course in Applied Cryptography*, Draft 0.5.

Camenisch, J. and Lysyanskaya, A. (2001) *An Efficient System for Non-Transferable Anonymous Credentials with Optional Anonymity Revocation*, Springer, Berlin, Heidelberg.

Chase, M. (2007) 'Multi-authority attribute based encryption', *Theory of Cryptography Conference*.

Chase, M. and Chow, S. (2009) 'Improving privacy and security in multi-authority attribute-based encryption', *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009*, 9–13 November, Chicago, Illinois, USA.

Chow, S. (2009) 'Removing escrow from identity-based encryption: new security notions and key management techniques', *Public Key Cryptography – PKC*, pp.256–276.

Deng, Y.Q. (2014) 'Dynamic attribute-based encryption scheme', *Computer Engineering & Science*, pp.312–325.

Dodis, Y. and Yampolskiy, A. (2004) 'A verifiable random function with short proofs and keys', in Vaudenay, S. (Eds.): *Public Key Cryptography – PKC 2005, Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, Vol. 3386.

Fan, C.I., Huang, S.M. and Ruan, H.M. (2014) 'Arbitrary-state attribute-based encryption with dynamic membership', *IEEE Transactions on Computers*, Vol. 63, No. 8, pp.1951–1961.

Fujisaki, E. and Okamoto, T. (1999) 'Secure integration of asymmetric and symmetric encryption schemes', *International Cryptology Conference*, pp.80–101.

Gennaro, R., Jarecki, S., Krawczyk, H. and Rabin, T. (2007) 'Secure distributed key generation for discrete-log based cryptosystems', *Journal of Cryptology*, Vol. 20, No. 1, pp.51–83.

Goyal, V., Pandey, O., Sahai, A. and Waters, B. (2006) 'Attribute-based encryption for fine-grained access control of encrypted data', *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, ACM, Alexandria, VA, USA, 30 October–3 November.

Green, M., Hohenberger, S. and Waters, B. (2011) 'Outsourcing the decryption of ABE ciphertexts', *Proceedings of the 20th USENIX conference on Security*.

Han, J., Susilo, W., Mu, Y., Zhou, J. and Au, M. (2015) 'Improving privacy and security in decentralized ciphertext-policy attribute-based encryption', *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 3, pp.665–678.

Hohenberger, S. and Waters, B. (2014) 'Online/offline attribute-based encryption', *International Workshop on Public Key Cryptography*.

Huang, L., Cao, Z., Liang, X. and Shao, J. (2008) 'Secure threshold multi authority attribute based encryption without a central authority', *International Conference on Cryptology in India: Progress in Cryptology*, pp.2618–2632.

Hui, M., Rui, Z., Wan, Z., Yao, L. and Lin, S. (2017) 'Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing', *IEEE Transactions on Dependable & Secure Computing*, Vol. 14, No. 6, pp.679–692.

Jarecki, S. and Liu, X. (2009) 'Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection', *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, Proceedings*, 15–17 March, San Francisco, CA, USA.

Jin, L., Huang, Q., Chen, X., Chow, S. and Xie, D. (2011) 'Multi-authority ciphertext-policy attribute-based encryption with accountability', *ACM Symposium on Information*.

Kan, Y. and Jia, X. (2012) 'Attributed-based access control for multi-authority systems in cloud storage', *IEEE International Conference on Distributed Computing Systems*.

Lai, J., Deng, R.H., Guan, C. and Weng, J. (2013) 'Attribute-based encryption with verifiable outsourced decryption', *IEEE Transactions on Information Forensics and Security*, Vol. 8, No. 8, pp.1343–1354.

Lewko, A. and Waters, B. (2011) *Decentralizing Attribute-Based Encryption*, Springer, Berlin, Heidelberg.

Li, J., Huang, X., Li, J., Chen, X. and Xiang, Y. (2014) 'Securely outsourcing attribute-based encryption with checkability', *IEEE Transactions on Parallel & Distributed Systems*, Vol. 25, No. 8, pp.2201–2210.

Lian, H., Wang, Q. and Wang, G. (2019) 'Large universe ciphertext-policy attribute-based encryption with attribute level user revocation in cloud storage', *International Arab Journal of Information Technology*, Vol. 17, No. 1, pp.107–117.

Lin, H., Cao, Z., Liang, X. and Shao, J. (2010) 'Secure threshold multi authority attribute based encryption without a central authority', *Information Sciences: An International Journal*, Vol. 180, No. 13, pp.2618–2632.

Ling, J., Chen, J., Chen, J. and Gan, W. (2021) 'Multiauthority attribute-based encryption with traceable and dynamic policy updating', *Security and Communication Networks*, Vol. 2021, No. 6, pp.1–13.

Liu, Z., Jiang, Z.L., Wang, X. and Yiu, S.M. (2018) 'Practical attribute-based encryption: outsourcing decryption, attribute revocation and policy updating', *Journal of Network & Computer Applications*, April, Vol. 108, No. C, pp.112–123.

Liu, H., Ping, Z., Chen, Z., Peng, Z. and Jiang, Z.L. (2017a) 'Attribute-based encryption scheme supporting decryption outsourcing and attribute revocation in cloud storage', *IEEE International Conference on Computational Science & Engineering*.

Liu, L., Wang, S. and Yan, Q. (2017b) 'A multi-authority key-policy ABE scheme from lattices in mobile ad hoc networks', *Ad-Hoc & Sensor Wireless Networks*, Vol. 37, Nos. 1–4, pp.117–143.

Liu, L., Wang, S., He, B. and Zhang, D. (2019) 'A keyword-searchable ABE scheme from lattice in cloud storage environment', *IEEE Access*, Vol. PP, No. 99, p.1.

Ma, S., Deng, R.H., Liu, S. and Qin, B. (2015) 'Attribute-based encryption with efficient verifiable outsourced decryption', *IEEE Transactions on Information Forensics & Security*, Vol. 10, No. 7, pp.1384–1393.

Naor, M., Pinkas, B. and Reingold, O. (2008) 'Distributed pseudo-random functions and KDCs', *Advances in Cryptology-Eurocrypt'99*, Vol. 1592, pp.327–346.

Ning, J., Cao, Z., Dong, X., Liang, K., Ma, H. and Wei, L. (2017) 'Auditable $\sigma$-time outsourced attribute-based encryption for access control in cloud computing', *IEEE Transactions on Information Forensics and Security*, Vol. 13, No. 1, pp.94–105.

Odelu, V., Kumar, A., Sreenivasa, Y., Kumari, S., Khan, M.K. and Choo, K.K.R. (2017) 'Pairing-based CP-ABE with constant-size ciphertexts and secret keys for cloud environment', *Computer Standards & Interfaces*, Vol. 54, pp.3–9.

Ostrovsky, R., Sahai, A. and Waters, B. (2007) *Attribute-Based Encryption with Non-Monotonic Access Structures*, DOI: 10.1145/1315245.1315270..

Pohoata, C. (2008) *Boole's Formula as a Consequence of Lagrange's Interpolating Polynomial Theorem*, arXiv.

Premkamal, P.K., Pasupuleti, S.K. and Alphonse, P. (2020a) 'Dynamic traceable CP-ABE with revocation for outsourced big data in cloud storage', *International Journal of Communication Systems*, No. 6, p.e4351.

Premkamal, P.K., Pasupuleti, S.K. and Alphonse, P.J.A. (2020b) 'Efficient escrow-free CP-ABE with constant size ciphertext and secret key for big data storage in cloud', *Int. J. Cloud Appl. Comput.*, Vol. 10, No. 1, pp.28–45.

Qian, X., Tan, C., Fan, Z., Zhu, W., Xiao, Y. and Cheng, F. (2018) 'Secure multi-authority data access control scheme in cloud storage system based on attribute-based signcryption', *IEEE Access*, Vol. 6, pp.34051–34074, DOI: 10.1109/ACCESS.2018.2844829.

Rahulamathavan, Y., Veluru, S., Han, J., Fei, L., Rajarajan, M. and Lu, R. (2016) 'User collusion avoidance scheme for privacy-preserving decentralized key-policy attribute-based encryption', *IEEE Transactions on Computers*, Vol. 65, No. 9, pp.2939–2946.

Ren, Y.J., Jian, S., Jin, W., Jin, H. and Lee, S.Y. (2015) 'Mutual verifiable provable data auditing in public cloud storage', Vol. 16, No. 2, pp.317–323.

Rivest, R.L., Shamir, A. and Adleman, L. (1978) 'A method for obtaining digital signatures and public-key cryptosystems', *Communications of the ACM*, Vol. 21, No. 2, pp.120–126.

Sahai, B.A. (2005) *Fuzzy Identity Based Encryption*, pp.457–473, Springer.

Shamir, A. (1979) 'How to share a secret', *Communications of the ACM*, November, Vol. 22, No. 11, pp.612–614.

Shamir, A. (1984) *Identity-Based Cryptosystems and Signature Schemes*, Vol. 196, Springer, Berlin, Heidelberg.

Taylor, N.E. and Ives, Z.G. (2006) 'Reconciling while tolerating disagreement in collaborative data sharing', *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 27–29 June, Chicago, Illinois, USA.

Waters, B. (2008) 'Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization', *International Workshop on Public Key Cryptography*.

Xu, X., Zhou, J., Wang, X. and Zhang, Y. (2016) 'Multi-authority proxy re-encryption based on cpabe for cloud storage systems', *Journal of Systems Engineering and Electronics*, February, Vol. 27, No. 1, pp.211–223.

Yang, K., Jia, X. and Ren, K. (2013) 'Attribute-based fine-grained access control with efficient revocation in cloud storage systems', *ACM SIGSAC Symposium on Information*, p.523.

Yu, S., Wang, C., Ren, K. and Lou, W. (2010) 'Attribute based data sharing with attribute revocation', *International Symposium on ACM Symposium on Information*, p.261.

Zhang, R., Wang, M., Hui, M. and Lin, S. (2015) 'Revisiting attribute-based encryption with verifiable outsourced decryption', *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 10, pp.2119–2130.