An Outsourced Multi-authority Attribute-Based Encryption for Privacy Protection with Dynamicity and Audit

Zhifa Deng[1] and Jiageng Chen[1,2(✉)]

[1] School of Computer Science, Central China Normal University, Wuhan, China
jiageng.chen@ccnu.edu.cn
[2] Wollongong Joint Institute, Central China Normal University, Wuhan, China

Abstract. Attribute-based Encryption (ABE) realizes a novel and practical many-to-many encryption paradigm, in which the encryptor can appoint someone to decrypt it, and the decryptor does not know who the encryptor is. Therefore, the ABE has better privacy preserving to a certain extent for both participants to a certain degree. The original attribute-based encryption scheme usually has only one trusted authority, and the state of the system is not flexible enough to meet various dynamic needs of users, which become the bottleneck of the system. In addition, the computational overhead of the decryption is not cheap. In order to reduce the cost of decryption, a series of schemes such as encryption/decryption outsourcing have been proposed. However, those solutions simply outsource the process independently without any in-depth investigation into the trustworthiness issues. Hence, we construct an outsourced multi-authority attribute-based encryption with dynamicity and auditing (OMADA-ABE), which makes the system more practical and more flexible. Our solution can support the dynamic changes of the system as well as the auditing of outsourced decryption information to solve the above-mentioned related problems, so as to meet the potential requirements related to the real-world system. What's more, we prove that the proposed scheme is secure against selective chosen-ciphertext attacks without random oracles, and it also achieves the collusion resistance. Finally, we compare our scheme with related researches and showed our advantage regarding the performance and other aspects.

Keywords: Multi-authority ABE · Dynamicity · Outsourcing decryption · Audit · Collusion resistance · Privacy protection

1 Introduction

Cloud computing is an emerging and popular business model, more and more data is stored in the cloud rather than in local data centers, which can also reduce the loss of system maintenance. Traditionally, data cloud servers are considered to be trusted to guarantee privacy security and confidentiality of data, and to enforce access control policies correctly. However, in the current era of big data, since the cloud service

W. Meng and W. Li (Eds.): BlockTEA 2022, LNICST 498, pp. 69–88, 2023.
https://doi.org/10.1007/978-3-031-31420-9_5

provider (CSP) and the double-end user are not in the same trusted domain, the storage platform is not directly controlled by the data owner, and the CSP may not be fully trusted, so the technology inevitably has some risks and it is no longer applicable. Therefore, protecting the privacy and security in the process of interacting with others is particularly important. In order to reduce users' concerns about data privacy, a common solution is to store data in the form of ciphertext, which can still ensure the security of user data even if the data server or storage device is corrupted. However, encrypted data also needs to be shared, and public key encryption or symmetric encryption lacks flexible access control. Therefore, a better way is attribute-based encryption (ABE), which was first proposed by Sahai and Waters [1] to protect the confidentiality of sensitive data. Attribute-based encryption is divided into ciphertext policy attribute-based encryption (CP-ABE) and key policy attribute-based encryption (KP-ABE), both of which can prevent unauthorized users from accessing data [2], even if users store data in untrusted servers.

Moreover, the expensive decryption overhead of CP-ABE is a serious drawback of the attribute-based encryption system, which may essentially prevent its widespread deployment (especially on some small devices or platforms). How to effectively solve this bottleneck is also a problem worth exploring. In addition, the complexity of access policy in ABE system will also increase the computational cost to a certain extent, making the pairing computation cost higher, so it is a very serious challenge to balance the policy complexity and reduce the computational cost. An effective method to reduce the user's decryption overhead is outsourcing decryption, which can outsource a large number of complex decryption operations to a third-party cloud server (CSP), so as to reduce the local overhead. However, CSP is considered untrustworthy third party, so how to check the correctness of outsourced decryption is a very challenging problem. Therefore, none of the above-mentioned research have focused on combining dynamicity and auditable mechanisms with the multi-authority setting.

1.1 Our Contribution

In our work, we studied the above interesting issues and further propose an OMADA-ABE (Outsourced Multi-Authority attribute-based encryption with Dynamicity and Audit) system scheme, which incorporates dynamicity into multi-authority ABE, provides both outsourcing and auditing while protecting user privacy.

The main contributions of this work can be summarized as follows:

- **Dynamic management in the multi-authority setting.** We realize a series of dynamic operations, such as free registration of AAs, free joining, leaving and updating of users under the setting of multiple authorities. The attribute authority (AA) is responsible for the management of attributes, and the CA manages each AA, so as to ensure the stability of the system while taking into account of the dynamic performance. The existence of multi-authorities will make the above operations more smoothly.
- **Self-Auditability (by AA and CA).** Different from the verifiable methods introduced in [3,4], our auditable method does not require any extra parameters to be

added to the ciphertext of our system, and it will not bring any additional computing cost to users. At the same time, there is no need to introduce the cost of third-party auditors, and the audit role is jointly undertaken by the AAs and the CA. Our scheme only needs to perform one inversion and decryption operation to check the correctness of cloud decryption, so this also makes our scheme more efficient and applicable.

1.2 Paper Organization

The overall organizational framework of this paper is as follows. In Sect. 2, we review a series of related works of attribute-based encryption, compare and analyze several types of attribute encryption with different features. In Sect. 3 and 4, we give the preliminaries including the related definitions and present the system security assumptions. In Sect. 5, we give the concrete scheme construction, and the security analysis of our scheme is presented in Sect. 6. In Sect. 7, an evaluation and comparison of relevant performance is provided. Finally, we conclude our work in Sect. 8.

2 Related Work

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, so a multi-authority attribute-based encryption system is proposed. Under a multi-authority ABE system, different attribute authorities manage different attribute sets and distribute corresponding attribute keys. In 2007, Chase implemented the Multi-authority Attribute-Based Encryption (MA-ABE) scheme for the first time [5]. In [5], the AA (Attribute Authority) are independent of each other and do not need to exchange information, but it needs a fully trusted CA (Central Authority) to manage all private information. What's more, it achieves collusion resistance by assigning a unique Global Identifier (GID) to each user. In 2008 Lin et al. [6] proposed an MA-ABE scheme that does not require a central authority. Later, Lewko and Waters [7] proposed a new MA-ABE scheme called decentralizing CP-ABE (DCP-ABE) scheme, in which the authority center CA is removed, and any party can become an authority.

Also, Green et al. [8] introduced outsourced decryption in the ABE system, so complex operations in the decryption phase can be outsourced to the ODCSP, and users can recover plaintext only by performing a power operation. Although complex operations are entrusted to the ODCSP, they do not consider the correctness and security of the cloud returned results, only obtain the decryption results. In order to solve the above problems, Ren et al. [9] proposed a mutually verifiable data audit method. Later, Lai et al. [10] introduced the verifiability of ABE to further verify the outsourcing results. Lin et al. [11] reviewed the verifiable outsourcing ABE and proposed a more effective construction method. In 2020 Sethi et al. [12] outsources expensive decryption operations to more powerful computing resources, reduces the overall cost and constrains third-party computing resources to be trusted.

Moreover, after the multi-authority attribute encryption is proposed, a series of functional features are also derived, such as policy updating or hiding [13], attribute revocation [14], searchable ABE [8], attribute-based encryption on lattices [15], online or

offline combined computing [16], large universe mechanisms [14], traceability scheme [12], verifiable scheme [3], and so on. However, only some papers are aimed at member management or auditing or outsourcing [9,11,17], just pay attention to one aspect, they cannot use multi-authority decentralized management while ensuring dynamicity. None of the above-mentioned research have focused on combining dynamicity and auditable mechanisms with multi-authority.

3 Preliminaries

3.1 Access Structures

Definition 1 ***(Access Structure*** [18]***).*** *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* $C \subseteq A$*. 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.,* $A \subseteq 2^{\{P_1,P_2,...,P_n\}} \setminus \{\emptyset\}$*. The sets in A are called the authorized sets, and the sets not in A are called the unauthorized sets.*

3.2 Lagrange Interpolation

According to [18,19], 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 $e : G \times G \rightarrow G_T$ 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 (DBDH)

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 $e : G_1 \times G_2 \rightarrow G_T$ 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 ϵ 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 2 ***(The decisional-BDH Assumption*** [20]***).*** *We say that the decisional BDH assumption holds if no polynomial-time adversary has non-negligible advantage in solving the decisional BDH problem.*

3.5 Trapdoor Function

We say a trapdoor is a secret that allows one to efficiently invert the function; however, if one do not know the trapdoor, it's still difficult to reverse the function.

Definition 3 ***(The trapdoor function scheme*** [21]***).*** *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, when we know the trapdoor, we can easily calculate the secret value.

4 Syntax and Security

4.1 Security Assumptions

In our scheme, we make some security assumptions as follows:

- The Central Authority (CA) must be fully credible and not collude with any other authority.
- The Attribute Authority (AA) is trusted, but it is not completely trusted, and it may be corrupted by other malicious adversary.
- The Data User (DU) is honest but curious, is likely to be under the profit-driven to collude with other users, thereby gaining unauthorized data access control.
- The CSP and OD-CSP are honest but curious, just like DU. They will faithfully perform their duties, but they may also have some curiosity about data.

4.2 Security Model

Here, we describe a selectively ciphertext policy and chosen-ciphertext attack (sCP-IND-CCA) model for the OMADA-CPABE scheme, played by a game between a challenger and an adversary. 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 [22,23].

- **Init**: The adversary A chooses a challenge access control policy T^* that he wants to attack and sends it to the challenger C.
- **Setup**: The challenger runs $CASetup()$ algorithm to output CA_{pk} and invokes $AASetup()$ algorithm to generate the required keys for corrupt and honest AA respectively. The adversary selects the corrupted authorities among all the AA, i.e. $S'_{AA} \subset S_{AA}$. Then the challenger gets public key AA_{pk} and secret key AA_{sk} for all AAs in S_{AA} by running $AASetup()$ algorithm. For all non-corrupted attribute authorities in $\tilde{S} : (S_{AA} - S'_{AA})$, only its public key AA_{pk} will be issued to A; Otherwise, the public/private key pairs AA_{pk} and AA_{sk} of the remaining corrupted AAs in S'_{AA} should be sent to the adversary A.
- **Query Phase 1**: In phase 1, the adversary A makes the following queries to challenger C as follows.
  1. Private key query. The adversary is allowed to request the private keys $SK_{u_{id}}$ corresponding to his/her desired attribute sets $S_1, ..., S_q$ from AAs and extract the private key $SK_{u_{id}}$. (However, there is a restriction that there must be at least one non-corrupted attribute authority AA in order to prevent the adversary from obtaining a sufficient number of secret keys). Finally, $SK_{u_{id}}$ will be recorded in a set L_{SK}.
  2. Decryption query. The adversary is also allowed to query the decryption of the given ciphertext CT and output the message M corresponding to the CT. If the ciphertext is not correctly constructed, then outputs $\perp$.
- **Challenge**: The adversary A submits (M_0, M_1, T^*) to the challenger, of which M_0 and M_1 is two messages of equal length and T^* is an access tree structure, and all the sets $S_1, ..., S_q$ in Phase 1 do not meet this access structure. Now the challenger randomly selects $b \in \{0, 1\}$ and $K \in G_T$, then computes the encryption result $E_K(M_b)$ of the message M_b and encrypts the symmetric key K under the access tree T^*, and finally generates the matching CT^*. Once there exits $SK_i \in L_{SK}$ that can decrypt $CT*$, then the challenge will be terminated. Otherwise, the ciphertext CT^* proceeds normally and is sent to the adversary.
- **Query Phase 2**: Repeat phase 1, more secret key queries can be made by the adversary when he satisfies the requirement given previously, but the only limitation is that decryption queries cannot be made again.
- **Guess**: Finally, the adversary has to guess which message the challenger encrypted, then outputs a guess b' of b. If $b' = b$, we say the adversary wins the game.

We now define the advantage of the adversary in the above game as follows,

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

Definition 4. *An Outsourced Multi-Authority Attribute-Based Encryption with Dynamicity and Audit scheme is said to be selectively ciphertext policy and chosen-ciphertext attack (sCP-IND-CCA secure) if for any polynomial-time adversary A, the advantage* $Adv^{CCA}_{OMADA-ABE}(A)$ *is negligible.*

5 Our Construction

Table 1. Notations

Notation	Meaning
S_{AA}	the set of all attribute authorities (AA)
S_{A_i}	the set of attributes maintained by AA_i
$S_{u_{id}}$	the set of attributes owned by user u_{id}
λ	the security parameter
U	the set of registered users
E_K	the symmetric encryption function with key K
D_K	the symmetric decryption function with key K
N_L	the set of all leaf nodes of access tree T
k_N	the threshold value associated with node N
d_N	the degree of the polynomial of node N
$parent_{(N)}$	the parent node of node N
$index_{(N)}$	the index associated with node N
$Attr(N)$	the attribute associated with node N
$val(N)$	the attribute value associated with node N

First, some symbol notations in this chapter are described in Table 1, and then the system design details of this paper are described in the following parts.

5.1 System Initialization

1. $\boldsymbol{CASetup(1^{\lambda}) \rightarrow ((CA_{sk}, CA_{pk}), (CA_{sig}, CA_{verify}))}$**.** The CA takes the security parameter λ as input, and then generates a bilinear mapping $e : G_1 \times G_2 \rightarrow G_T$, where G_T is a multiplicative group, G_1 and G_2 are two additive groups. 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 groups with prime order q. The CA perform the steps as follows:

- Step 1: Choose two random elements α, β from Z_q^*.
- Step 2: Selects two one-way collision-resistance hash functions, named H_1 and H_2:
  $H_1 : \{0,1\}^* \rightarrow {Z_q}^*$
  $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_\lambda}$ (where l_λ represents the string length.)
- 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: Next, the CA exploits cryptography algorithm to generate a pair of signature verification key (CA_{sig}, CA_{verify}), and assists AA to verify the user's identity after the keys are generated.

After the subsequent AASetup phase is deployed, the CA still needs to initialize some parameter settings in the systems. For details, see the AAsetup phase.

2. $\boldsymbol{AAEnroll(CA_{sk}, CA_{pk}, AA_{\mathrm{inf}\,o}) \rightarrow (AA_i, CA_{verify}, CA_{sk})}$**.** Each authority sends a request to CA to register to the system. The CA runs the algorithm and generates a global unique identity for each legitimate authority in the system, and sends its private key CA_{sk} and verify key CA_{verify} to every legal AAs. Therefore, the AA has the ability to use the authentication key Ca_{verify} to check whether the user is valid. The AA will cooperate with the user to complete the verification once they have relevant requirements.

3. $\boldsymbol{AASetup(CA_{sk}, AA_i) \rightarrow (AA_{i,sk}, AA_{i,pk})}$**.** Every AA runs the algorithm with CA_{sk} as input, and then outputs a pair of public and secret keys as follows:

– Step 1: The 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, the CA then sets two default users in this step to be used as the credential information to bind the user and AA together. The default user generation process is as follows:

(1) First, set $U = \{0, 1\}$ and randomly select some values $\{v_{u_{id},j}\}_{\forall u_{id} \in U, j \in S_{A_i}}$ and $\{\sigma_i\}_{\forall i \in S_{AA}}$ in Z_q^*.

(2) Then, we get

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

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

4. $\boldsymbol{UserEnroll(CA_{sk}, CA_{pk}, u_{\mathrm{inf}\,o}) \rightarrow (u_{id}, cert(u_{id}), u_{sk}, u_{pk})}$**.** The algorithm randomly picks a $t_{u_{id}} \in Z_p$, then outputs the public and secret key pair of the user $\{u_{pk}, u_{sk}\}$ as follows:

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

Meanwhile, it 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).

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

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

1. $\boldsymbol{KeyGen(CA_{pk}, AA_{i,sk}, u_{id}, S_{u_{id}}) \rightarrow (SK_{u_{id}})}$**.** Each user registered to the system has its own global identity. They request the decryption key from the AA bound to their own attributes.

- Step 1: Randomly pick some elements $\{h_{u_{id},j}, \{\rho_i, \sigma_i\}_{\forall i \in S_{AA}}, \{v_{u_{id},j}\}_{\forall j \in S_{u_{id}} \bigcap S_{A_i}}, \{r_{i,j}\}_{\forall i \in S_{AA} \forall j \in S_{u_{id}} \bigcap S_{A_i}}\}$ in Z_q^*.
- Step 2: Compute

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

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

2. $\boldsymbol{Update(u_{id}, j, m'_{u_{id},j}) \rightarrow (SK_{u_{id},j})}$**.** With the dynamic changes of users' identities, roles, capabilities and other requirements in the system, their attributes are constantly changing. Therefore, users can update their corresponding attribute values according to the current actual scenario, such as updating their j^{th} attribute value to ${m'}_{u_{id},j}$ to meet the needs of 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

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

  to user u_{id}.

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

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

3. $\boldsymbol{Revoke(u_{id})}$**.** It is guaranteed that the leaving user cannot legally exist in the system, both the certificate and the private key are revoked. What's more, the credentials of system will also be modified, and his/her all related information will be removed. The algorithm update $\{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j \in S_{A_j}}\}$ as follows:

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

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

5.3 Encryption and Decryption

1. $\boldsymbol{Encrypt(CA_{pk}, AA_{i,pk}, T, K, M) \rightarrow (CT).}$ In our study, due to the encryption algorithm is operated under the tree access structure T [24], the generating process of ciphertext is divided into two parts: leaf node and internal node.

a) **Access tree structure construction** [25]**:** 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.

Then the generated ciphertext is below:

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

where $V_{Att(N)}$ are the credential information of current attribute node N, $H_2(val(N))$ are the hash value of the attribute information corresponding to the current node N.

2. $\boldsymbol{Decrypt_{ODCSP}(CT, SK_{u_{id}}) \rightarrow (K_1).}$ After the user gets the ciphertext CT from the CSP, since there is a certain exponential operation for ciphertext decryption, our scheme is tend to transpose the overhead operation to the outsourcing decryption CSP.

If N is a leaf node: Then we let $j = Att(N)$ and $H_2(val(N)) = v_j$. If $j \in S_{A_i}$, then:

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

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

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

If N is an internal node: It invokes the algorithm $DecryptNode(CT, SK_{u_{id}}, N_c)$ and stores its output results as F_{N_c}, where nodes N_c are children of N. Let S_N be an arbitrary k_n-sized set of child nodes N_c such that $F_{N_c} \neq \perp$. If there is no such set then the node does not meet the requirements, and the function returns $\perp$.

Otherwise, we get

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

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

Then, the algorithm calls the function on the root node R of the tree T. If the set S satisfies the access tree, we set $A = DecryptNode(CT, SK_{u_{id}}, R)$, 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 calculated from the formula $\tilde{C} \cdot A/(e(C, D_{u_{id}}) \cdot C')$ where

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

5.4 Audit

1. $\boldsymbol{Audit_{AA}(AA_{i,sk}, CT, Decrypt_{ODCSP}(CT, SK_{u_{id}})) \rightarrow (1/0).}$ 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 K_1 and $\bar{M}$ are consistent and unified, so the preliminary audit 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.

2. $\boldsymbol{Audit_{CA}(Audit_{AA}(*) == 1, CA_{pk}, CT, K, M) \rightarrow (1/0).}$ The algorithm will judges 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. 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; otherwise, it indicates that there is collusion between the outsourcing decryption CSP and AA, or at least one party has calculation error.

5.5 Decryption by User

$\boldsymbol{Decrypt_{DU}(Audit_{CA}(*) == 1, cert(u_{id}), CT, K_1, K_2) \rightarrow (M/\perp).}$ The algorithm will judges 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 $E_K(\bar{M}) = M$.

6 Security Analysis

Theorem 1. *The OMADA-ABE scheme is sCP-IND-CCA secure under the DBDH assumption.*

Proof. Assuming that there exits an adversary in polynomial probability time that can break our algorithm, then there will be a challenger that can break the DBDH assumption by using the adversary. 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 has been 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: Specify a DBDH instance$(G_1, G_2, G_T, q, g, h, e, ag, bg, cg, ah, bh, ch, Z)$, the challenger generates the system default user list $U = \{0, 1\}$ and $\{V_j, \{\bar{v}_{u_{id},j}\}_{\forall j \in S_{A_i}}\}$ according to the $AASetup()$ algorithm in Sect. 5.1 initially. In addition, 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 secret key $SK = \{\alpha, \beta\}$. The simulator B_e receives the above DBDH instance generated by the challenger and other relevant parameters, the simulator is ready now. Since this paper assumes the model as selective, the adversary can adaptively choose a challenge access policy tree T^* and send it to the simulator. The simulator builds a list L_{SK}, and starts to simulate the oracles in Phase 1.

Setup: The challenger runs $CASetup()$, $AAEnroll()$ and $AASetup()$ algorithm. 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 sends the public key of the corrupted authority together with the private key to the adversary. Now the adversary A can get the $SK_{u_{id},k}(D_{u_{id}}, D_{u_{id},k}, D'_{u_{id},k}, D''_{u_{id},k})$ alone for u_{id}. For the remaining AAs that are not corrupted, only the public key is given to the adversary.

Phase 1: The simulator B_e generates private keys for system default users $SK_{u_{id}}$ with complete set and satisfying access policy by following $KeyGen()$ algorithm. Then the adversary A queries for the private key of the i^{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}. Upon receiving a decryption request from the adversary, 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 will check whether a private key $SK_{u_{id},i}$ used to decrypt CT exists in L_{SK}. If there exists key in L_{SK} that can decrypt, B_e decrypts CT exploiting $SK_{u_{id},i}$ and then returns the final result to A. If not, then the simulator B_e will create a pseudo user w and make full use of the user $w's$ private key to decrypt. 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

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

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:

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

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

Challenge: At the beginning of the challenge, the adversary selects two messages of equal length M_0 and M_1 in advance, and sends them to the simulator as soon as he receives all the messages from the simulator. After receiving the (M_0, M_1, T^*), the simulator B_e creates a set list L_C to store the values of the nodes in the subsequent access tree. Meanwhile, let R be the root node of the received access policy tree T^*. Then set $m = val(N)$, $j = att(N)$, and get $C_N = v_{att(N)} C_{0_{\forall N \in N_L}}$, $C'_N = q_N(0) H_2(val(N))$, $C''_N = H_2(val(N)) v_{att(N)}$. Now we store some values of the challenged node $\{C_N, C'_N, C''_N\}_{N \in N_L}$ in the list L_C. The B_e randomly picks a $b \in \{0,1\}$ and $K' \in G_T$, here we treat the selected message as M_b, and the b takes the value 0 or 1. After multiple operations above, we calculate the ciphertext $CT^* = \{T^*, \tilde{C} = ZK_1, C = \gamma g, C' = \frac{Z}{e(\alpha g, \gamma h)}, \bar{M}_b = E_{K'}(M_b), C_F = F(x_i, K_1 \oplus \bar{M}_b), \{C_N, C'_N, C''_N, \{\bar{v}_{u_{id},Att(N)}\}_{\forall u_{id} \in U}\}_{N \in N_L}\}$, where the encrypted value of the node is retrieved from the encrypted list L_C. In addition, we emphasize that CT^* is regarded as a valid encryption information for M_b when $Z = e(g,h)^{abc}$; Otherwise, the CT^* is just a random value. The challenge will be terminated if the key $SK_{u_{id},k} \in L_{SK}$ is found that can be used to decrypt the CT. If not, the B_e returns the result(CT^*) to A.

Phase 2: The adversary continues to follow phase 1 to make selective queries as he did, except for all decryption queries.

Guess: Finally, after several rounds of interaction, the adversary A gets his guess b'. The B_e outputs 1 and guess that $Z = e(g,h)^{abc}$(a valid BDH-tuple) if $b' = b$. Otherwise, the adversary is wrong and B_e outputs 0 denote that the Z is not a valid combination of elements but a random value in G_T. Therefore, if a polynomial time adversary can break our scheme with non-negligible advantage at least ε, which means that a challenger has non-negligible ε' to break the DBDH assumption where $\varepsilon' \geq \varepsilon - \delta$ and δ is a negligible advantage. Also, we can obtain

$\Pr[B_e(g,h,ag,bg,cg,ah,bh,ch,e(g,h)^{abc}) = 1] = \Pr[b' = b]$ if $Z = e(g,h)^{abc}$, where $\left|\Pr[b' = b] - \frac{1}{2}\right| \geq \varepsilon$. Otherwise(the Z is chosen at random in G_T), we have $\Pr[B_e(g,h,ag,bg,cg,ah,bh,ch,Z) = 1] = \Pr[b' = b]$, where $\left|\Pr[b' = b] - \frac{1}{2}\right| \leq \delta$ (δ is one probability that guarantees semantic security under symmetric encryption E_K). Finally, we have

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

7 Comparisons

In this section, we mainly compare the function and performance analysis of previous research schemes.

Table 2. Comparison of security and functionality

Schemes	CA	MA	OD	AU	DN	CR	PP	GID	MC	Hardness	Model
[23]	×	√	×	×	×	√	√	Privacy	CPA	DBDH	ROM
[19]	√	×	√	×	√	√	√	privacy	CPA	DBDH	ROM
[26]	×	√	×	×	×	×	√	Public	CPA	q-PBDHE q-SDH	ROM
[27]	√	×	×	×	√	×	×	Public	CPA	q-BDHE	ROM
[28]	×	√	×	×	×	√	√	Privacy	CPA	DBDH	ROM
[13]	√	×	×	×	√	×	√	Privacy	CCA	DBDH	Standard
[29]	√	√	×	×	×	√	√	Public	CPA	q-PBDHE	ROM
[30]	√	×	√	×	×	√	√	Private	CPA	q-BDHI	ROM
Our scheme	√	√	√	√	√	√	√	Privacy	CCA	DBDH	Standard

Abbreviated symbol description: **MA** = Multi-Authority, **OD** = Outsourced Decryption, **AU** = Auditability, **DN** = Dynamicity, **CR** = Collusion Resistance, **PP** = Privacy Protection, **GID** = Global Identity, **MC** = Message Confidentiality.

7.1 Feature and Security Comparisons

A detailed comparison of some relevant features between the proposed scheme and part of the existing schemes is given in Table 2. Obviously, some challenging features like traceable, policy updating and data access limited have been addressed in schemes [19,29]. But we have also addressed the perfect combination of dynamicity and multi-authorities, making the system practical while maintaining flexibility. In addition, we apply the auditing feature to solve the verification of outsourcing decryption to ensure the correctness for decrypted ciphertext by ODCSP.

7.2 Computational Analysis

In order to unify the measurement scale, this study divides the above schemes listed in the comparison into two categories according to the number of attribute authority: multi-authority schemes and non-multi-authority(single CA), and makes a horizontal comparison.

We give a description of the relevant symbols involved in the comparison process in Table 3. Among them, Table 4 represents the comparison of schemes without multi-authority, and Table 5 represents another.

Table 3. Notations used in performance evaluation.

Notation	Meaning
t_p	the time of one pairing operation
t_e	the time of one exponentiation operation
n_u	the number of users attributes
n_a	the number of attribute authorities
n_τ	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

As the results shown in Table 4, the key generation cost of our scheme is lower than other series schemes, which is also better than Fan et al. [13], because our scheme requires fewer times of exponentiation operations. In Table 5, the DU in our scheme does not require any exponentiation or pairing operations in the decryption operation, which greatly reduces the decryption cost of client, and the user only needs to complete a symmetric decryption once. We also find that the client-side decryption overhead is much better than schemes with outsourced decryption, such as [19] and [30]. In this study, the decryption cost of DU is outsourced to the ODCSP, and the computational cost of this part is $(3n_\tau + 4)t_p + n_{nl}t_e$. Therefore, the total decryption cost can be calculated by combining the two results. Compared with other outsourced schemes [19, 30], our OMADA-CPABE scheme requires less computation time. In Table 4, it is not difficult for us to find that the computation overhead in all these schemes increases linearly with the increase of the number of members and attributes.

Table 4. Comparisons of computation overhead(a)

Schemes	Key Generation	Encryption	Decryption(user)	Outsourced Decryption
[23]	$({n_a}^2+1)t_e$	$(n_\tau+2)t_e$	$(n_a+3)t_p+t_e$	-
[26]	$10n_at_e$	$(n_a+3)t_e+n_ct_p$	$(4n_u+2l)t_p$	-
[28]	$5n_ct_e$	$2n_ct_e+n_ct_p$	$(n_c+n_a+1)t_p$	-
[29]	$2n_ut_e$	$(n_c+1)t_p+3n_ct_e$	$5n_c(t_p+t_e)$	-
Our scheme	1	$(n_c+1)t_e+t_p$	1	$(3n_\tau+4)t_p+n_{nl}t_e$

Table 5. Comparisons of computation overhead(b)

Schemes	Key Generation	Encryption	Decryption(user)	Outsourced Decryption
[19]	$(n_u+3)t_e$	$(n_\tau+2)t_e$	t_e	$(n_u+2)t_p+n_{nl}t_e$
[27]	$(3+n_u)t_e+2t_H$	$(n_c+1)t_e+t_p$	$(2n_u+1)t_p$	-
[13]	1	$(2n_c+1)t_e+t_p$	$(n_u+1)t_p+n_{nl}t_e$	-
[30]	$(n_u+5)t_e$	$(5n_\tau+2)t_e$	t_e	$(3n_\tau+3)t_p+lt_e$
Our scheme	1	$(n_c+1)t_e+t_p$	1	$(3n_\tau+4)t_p+n_{nl}t_e$

7.3 Storage Overhead

In this part, we only consider the storage overhead of public parameters, private keys, and ciphertext. The Table 6 combines several kinds of attribute-based encryption schemes in the past and gives a comparison of the storage required. For the cost storage of public parameters, the proposed scheme is consistent with the storage cost of [13] and [23]. Also, it can be clearly seen from Table 6 that the private key storage size of our scheme is slightly lower than that of other schemes, which can benefit the storage space to some extent. However, the ciphertext size in our work still has room for further optimization. How to balance the storage cost and security efficiency is also a problem that needs to be weighed and considered to ensure the high efficiency and security of the whole system.

Table 6. Comparisons of Storage

Schemes	Size of Public Parameters	Size of Private Key	Size of Ciphertext
[23]	$\lvert G_1\rvert+\lvert G_2\rvert+\lvert G_T\rvert$	$(n_u+1)\lvert G\rvert$	$(n_c+2)\lvert G\rvert$
[19]	$\lvert G\rvert+\lvert G_t\rvert$	$(n_u+3)\lvert G\rvert$	$(n_c+1)\lvert G\rvert+\lvert G_t\rvert$
[26]	$2\lvert G\rvert+\lvert G_T\rvert$	$6n_u\lvert G\rvert+\lvert G_T\rvert$	$(2l+3)\lvert G\rvert+\lvert G_T\rvert$
[27]	$2\lvert G\rvert$	$(n_u+2)\lvert G\rvert$	$(l^2+2)\lvert G\rvert$
[28]	$2\lvert G_1\rvert+\lvert G_2\rvert$	$3n_u\lvert G_1\rvert$	$(2n_c+1)\lvert G_1\rvert+\lvert G_2\rvert$
[13]	$\lvert G_0\rvert+\lvert G_1\rvert+\lvert G_T\rvert$	$3n_u\lvert G_0\rvert+\lvert G_1\rvert$	$2\lvert G_0\rvert+(3n_c+1)\lvert G_T\rvert$
[29]	$3\lvert G\rvert+\lvert G_T\rvert$	$(4n_u+1)\lvert G\rvert$	$(5l+1)\lvert G\rvert+\lvert G_T\rvert$
[30]	$2\lvert G_T\rvert$	$(2n_u+4)\lvert G_T\rvert$	$(3l+1)\lvert G_T\rvert$
Our scheme	$\lvert G_1\rvert+\lvert G_2\rvert+\lvert G_T\rvert$	$3\lvert G_1\rvert+n_u\lvert G_2\rvert$	$n_c\lvert G_2\rvert+\lvert G_T\rvert$

8 Conclusion

The scheme in the study realizes an instance of the combination of dynamicity and multi-authority, then adds dynamic management and outsourcing decryption method with auditing under multi-authority, which optimizes the performance bottleneck of encryption and decryption, making attribute-based encryption more practical and flexible. In this study, the high linear pairing cost generated in the decryption process is outsourced to the cloud server, and the content decrypted by the server is audited twice to ensure the integrity and correctness of the information and the security of the system. All of the above advantages make ABE more efficient and flexible in practical application. However, to be honest, our work doesn't consider the size of the ciphertext, only focus on security and encryption efficiency of the whole system. Therefore, in the subsequent research, the ciphertext size is compressed to optimize the overall performance and further enhance the robustness of the system while ensuring user privacy protection and security.

Acknowledgement. This work has been partly supported by the Fundamentral Research Funds for the Central Universities (No. 30106220482).

References

1. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security & Privacy (2007)
3. Qin, B., Deng, R.H., Liu, S., Ma, S.: Attribute-based encryption with efficient verifiable outsourced decryption. IEEE Trans. Inf. Forensics Secur. **10**, 1384–1393 (2015)
4. Hui, M., Rui, Z., Wan, Z., Yao, L., Lin, S.: Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing. IEEE Trans. Dependable Secure Comput. **14**(6), 679–692 (2017)
5. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_28
6. Huang, L., Cao, Z., Liang, X., Shao, J.: Secure threshold multi authority attribute based encryption without a central authority. In: International Conference on Cryptology in India: Progress in Cryptology, pp. 2618–2632 (2008)
7. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_31
8. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: Proceedings of the 20th USENIX Conference on Security (2011)
9. Ren, Y.J., Jian, S., Jin, W., Jin, H., Lee, S.Y.: Mutual verifiable provable data auditing in public cloud storage. éè · è **16**(2), 317–323 (2015)
10. Lai, J., Deng, R.H., Guan, C., Weng, J.: Attribute-based encryption with verifiable outsourced decryption. IEEE Trans. Inf. Forensics and Secur. **8**, 1343–1354 (2013)
11. Lin, S., Zhang, R., Ma, H., Wang, M.: Revisiting attribute-based encryption with verifiable outsourced decryption. IEEE Trans. Inf. Forensics Secur. **10**(10), 2119–2130 (2015)

12. Sethi, K., Pradhan, A., Bera, P.: Practical traceable multi-authority CP-ABE with outsourcing decryption and access policy updation. J. Inf. Secur. Appl. **51** (2020)
13. Fan, C.I., Huang, S.M., Ruan, H.M.: Arbitrary-state attribute-based encryption with dynamic membership. IEEE Trans. Comput. **63**(8), 1951–1961 (2014)
14. Lian, H., Wang, Q., Wang, G.: Large universe ciphertext-policy attribute-based encryption with attribute level user revocation in cloud storage. Int. Arab J. Inf. Technol. **17**(1), 107–117 (2019)
15. Liu, L., Wang, S., Yan, Q.: A multi-authority key-policy ABE scheme from lattices in mobile ad hoc networks. Ad-hoc Sens. Wirel. Netw. **37**(1–4), 117–143 (2017)
16. Hohenberger, S., Waters, B.: Online/offline attribute-based encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 293–310. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_17
17. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: International Symposium on ACM Symposium on Information, p. 261 (2010)
18. Xu, X., Zhou, J., Wang, X., Zhang, Y.: Multi-authority proxy re-encryption based on CPABE for cloud storage systems. J. Syst. Eng. Electron. **27**, 211–223 (2016)
19. Premkamal, P.K., Pasupuleti, S.K., Alphonse, P.: Dynamic traceable CP-ABE with revocation for outsourced big data in cloud storage. Int. J. Commun. Syst. (6), e4351 (2020)
20. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
21. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.5 (2020)
22. Kan, Y., Jia, X.: Attributed-based access control for multi-authority systems in cloud storage. In: IEEE International Conference on Distributed Computing Systems (2012)
23. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, 9–13 November 2009 (2009)
24. Sarma, R., Kumar, C., Barbhuiya, F.A.: MACFI: a multi-authority access control scheme with efficient ciphertext and secret key size for fog-enhanced IoT (2021)
25. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. ACM (2006)
26. Han, J., Susilo, W., Mu, Y., Zhou, J., Au, M.: Improving privacy and security in decentralized ciphertext-policy attribute-based encryption (2015)
27. Deng, Y.Q.: Dynamic attribute-based encryption scheme. Comput. Eng. Sci. (2014)
28. Rahulamathavan, Y., Veluru, S., Han, J., Fei, L., Rajarajan, M., Lu, R.: User collusion avoidance scheme for privacy-preserving decentralized key-policy attribute-based encryption. IEEE Trans. Comput. **65**(9), 2939–2946 (2016)
29. Ling, J., Chen, J., Chen, J., Gan, W.: Multiauthority attribute-based encryption with traceable and dynamic policy updating. Secur. Commun. Netw. **2021**(6), 1–13 (2021)
30. Ning, J., Cao, Z., Dong, X., Liang, K., Ma, H., Wei, L.: Auditable σ-time outsourced attribute-based encryption for access control in cloud computing. IEEE Trans. Inf. Forensics Secur. **13**(1), 94–105 (2017)