Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.039170
Article

A Lightweight ABE Security Protection Scheme in Cloud Environment Based on Attribute Weight

Lihong Guo*, Jie Yang and Haitao Wu

Department of Information and Communications Engineering, Nanjing Institute of Technology, Nanjing, 211167, China
*Corresponding Author: Lihong Guo. Email: guolihongnj@163.com

Received: 12 January 2023; Accepted: 05 June 2023; Published: 30 August 2023

Abstract: Attribute-based encryption (ABE) is a technique used to encrypt data, it has the flexibility of access control, high security, and resistance to collusion attacks, and especially it is used in cloud security protection. However, a large number of bilinear mappings are used in ABE, and the calculation of bilinear pairing is time-consuming. So there is the problem of low efficiency. On the other hand, the decryption key is not uniquely associated with personal identification information, if the decryption key is maliciously sold, ABE is unable to achieve accountability for the user. In practical applications, shared message requires hierarchical sharing in most cases, in this paper, we present a message security hierarchy ABE scheme for this scenario. Firstly, attributes were grouped and weighted according to the importance of attributes, and then an access structure based on a threshold tree was constructed according to attribute weight. This method saved the computing time for decryption while ensuring security and on-demand access to information for users. In addition, with the help of computing power in the cloud, two-step decryption was used to complete the access, which relieved the computing and storage burden on the client side. Finally, we simulated and tested the scheme based on CP-ABE, and selected different security levels to test its performance. The security proof and the experimental simulation result show that the proposed scheme has high efficiency and good performance, and the solution implements hierarchical access to the shared message.

Keywords: Attribute-based encryption; cloud security; message hierarchy; attribute weight; access control

1 Introduction

The development of big data, cloud environment, and multi-user ecological environment makes the traditional access control and encryption system no longer applicable, and ABE came into being. ABE is an extension of the traditional public key encryption system, and it was originally proposed by Sahai and Waters in 2005. ABE is a new way to achieve access control on ciphertext, it has access control capability and enhances the flexibility of access control, and has been applied in building a variety of security systems.

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

ABE is different from conventional access control, its characteristics are as follows: 1) the information processed in the ABE is always stored in ciphertext, and they can be used in a completely dangerous hostile environment, even if attacked, and the information will not be obtained. 2) The data owner encrypts information according to attributes, and it restricts the users' access and decryption capabilities to ciphertext through access policies, ensuring the flexibility of data sharing. 3) In ABE, the user keys are related to the random polynomials, and it effectively prevents collusion attacks.

At present, ABE is widely used in the field of access control, which greatly guarantees the confidentiality, integrity, availability, and sharing security of data in cloud storage. However, there are still the following disadvantages: 1) Low efficiency: A large number of bilinear mappings are used in ABE, and it is time-consuming; 2) No accountability: The decryption key is not uniquely associated with personal identification information. Since multiple users share the same attributes, if the decryption key is maliciously sold, ABE is unable to achieve accountability for the user; 3) Heavy burden on users: The general ABE scheme has a large amount of computation, and the authorized user downloads the ciphertext to the local for decryption, which is a burden for the user. 4) Ignore the hierarchy of shared messages: Most of the current solutions focus on access control, but the shared messages have the multilevel hierarchy characteristic, the hierarchy structure hasn't been explored in ABE.

Given the above problems in ABE, in this work, we provide a message security hierarchy ABE scheme based on attribute weight, which not only considers the efficiency, accountability, and the user's computing power but also considers the fine-grained access needs of the message itself.

Its main contributions are as follows:

(1) Propose a new ABE scheme for the scenario where the shared message requires hierarchical access.
(2) Construct an access structure based on a threshold tree and attribute weight.
(3) Provide a two-stage decryption approach. It greatly reduces the burden on users.
(4) Design a series of experiments to test its performance.

Organization: The rest of the paper is organized as follows. Section 2 is the related work. Section 3 is the preliminaries. Section 4 describes the construction, and then the implementation and performance are shown in Section 5. Finally, we conclude in Section 6.

2 Related Work

2.1 Related Research of ABE

In literature [1], Shamir first proposed Identity-Based Encryption (IBE), it is assumed that there is a trusted key generation center to perform the work of issuing and verifying the legal identity of users. Later, people use biometrics to represent IBE's functions, but existing IBE cannot be used due to noise issues, so in 2005, Sahai and Waters proposed the vague concept of IBE, named Fuzzy Identity-Based Encryption (FIBE) [2,3], and this paper proposed the concept of ABE for the first time. In FIBE, the identity is regarded as a set of attributes, the idea of a threshold scheme is introduced, and multiple public keys of users are constructed into a threshold structure with a logical relationship. It has strong fault-tolerance and anti-collusion attack abilities.

As the research goes deeper, Sahai and Waters further clarified and divided the ABE into Key Policy ABE (KP-ABE) and Ciphertext Policy ABE (CP-ABE). In 2007, Bethencourt, Sahai, and Waters proposed attribute encryption based on ciphertext strategy for the first time [4] (named CP-ABE). In this scheme, the structure of the access tree was used to hide the key of source data. The attributes and attribute values set by the leaf node for the data owner, as well as the secret values passed to the node by the parent node, are encrypted and processed. Users need to have the specific attributes of the leaf node to get the secret value S of this node. The non-leaf node is the threshold node, and the user needs to have the minimum attribute set that meets the threshold to decrypt the secret value of the threshold node. CP-ABE has a wide application that is generally used for encrypted data storage and fine-grained sharing on public clouds. The schemes or models based on CP-ABE include the CP-ABE model of the hidden access structure, ABE scheme with multiple authorization centers, ABEs that can be held accountable: white-box ABEs and black-box ABEs, etc. [5]. In KP-ABE scheme [6], ciphertexts are associated with sets of attributes, and private keys are associated with access structures.

Next, in 2012, Bonch proposed the concept of Functional Encryption (FE) [7], FE is an extension of traditional encryption, which allows a third party to calculate the function value of the output plaintext without decrypting the ciphertext. In literature [8], Garg used indistinguishable obfuscator early enough for general FE; later, many schemes improved the efficiency of FE: Abdalla first proposed a public key-based inner product function encryption scheme, and Boneh proposed the Order Revealing Encryption (ORE) system. Deng proposed Process Based Encryption (PBE) [9].

At present, almost all algorithms of ABE focus on access control of the user [10,11], but the decryption process of the ABE algorithm is coarse-grained control, and the user can either decrypt it, or cannot decrypt it at all. In practical applications, we always focus on the access control of the files, and need to perform fine-grained control over the decryption process, in recent years, some scholars have shifted the focus to access control of the shared messages, because the message is the purpose of users.

In 2012, Hierarchical Attribute-Set-Based Encryption (HASBE) was proposed by extending CP-ABE scheme with a hierarchical structure. This scheme employs multiple value assignments for access expiration time to deal with user revocation more efficiently than existing schemes. In 2013, Rouselakis and Waters proposed two large-scale ABE structures. This is a new large-scale ciphertext policy attribute encryption scheme on first-order bilinear groups, and significantly improves the efficiency of the KP-ABE system. It brings "program and cancel" techniques to address this problem, aiming to provide practical large-scale implementations of attribute encryption.

In 2016, File Hierarchy CP-ABE (FH-CP-ABE) [12] scheme is proposed. In this scheme, at different access levels, the shared messages are divided into many hierarchy subgroups, and the layered access structures are integrated and encrypted with the integrated access structure. The messages in the same hierarchical structure could be encrypted by an integrated access structure. In 2021, Extended File Hierarchy CP-ABE (EFH-CP-ABE) scheme [13] is proposed, which can encrypt multiple files on the same access level in the existing FH-CP-ABE scheme. It greatly saves storage space and computation costs on the cloud servers.

2.2 The Existing Problem

Throughout the development process of ABE, it has experienced a series of upgrades and improvements [14–17]. But access control for shared files is popular from a practical standpoint, especially since it is suitable for those big institutions or companies which have many hierarchical sectors.

At present, although the existing file hierarchy ABE schemes have saved time for encryption and decryption, and reduced the burden of storage, there are still some important factors not taken into account. For example, 1) Lack of accountability. If the decryption key is maliciously sold, ABE is unable to achieve accountability for the user. 2) Low efficiency. At present, the biggest burden of ABE is on the user. When the ciphertext files become larger and larger, ciphertext files must be downloaded to the user for decryption, which meets a challenge to the storage and computing power of the user. All of these are lack of in the above file hierarchy ABE. So, in this paper, we propose a new message hierarchy ABE scheme.

In this work, a new ABE scheme was proposed for the application scenario where the shared message requires hierarchical access. It makes up for the shortcomings of previous ABE programs.

3 Basic Theoretical Knowledge

To elaborate the proposed scheme in this paper, some relevant knowledge will be introduced.

3.1 Hash Function

A hash function is any function that converts an input of any length into a fixed-length output, and the output is the hash value. The values are called hash values, hash codes, checksums, or simply hashes. This transformation is a compression map. Some famous hash algorithms are MD5, SHA-256, SHA-512, et al.

3.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is the theoretical basis of bilinear pairing. Generally, an elliptic curve refers to the curve determined by the Weierstrass equation. In cryptography, the elliptic curve on the finite field is used, that is, the variables and coefficients are all elements on the finite field, and the elliptic curve satisfies Eq. (1):

$$y^2 = x^3 + ax + b \pmod p \tag{1}$$

All points (x, y) satisfied the Eq. (1), plus an infinite point O constitutes a set, where a, b, x, y are all on the finite field GF(p), and p is a prime number. Here the elliptic curve is denoted as: $\mathrm{E}_p(a, b)$. The elliptic curve has a finite number of points.

While $4a^3 + 27b_2 \neq 0$, based on the set $\mathrm{E}_p(a, b)$, it can define an Able groups. Assume that: $P, Q \in \mathrm{E}_p(a, b)$, then: $P + \mathrm{O} = P$; If P = (x, y), then (x, y) + (x,−y) = O, that point (x,−y) is the additive inverse of P's, which means −P.

Assume that P = (x1, y1) and $Q = (x_2, y_2)$, P is not equal to Q, $S = P + Q = (x_3, y_3)$, the value of S is determined by the following rules:

$$x_3 \equiv \lambda^2 - x_1 - x_2 \pmod p$$

$$\mathrm{y}_3 \equiv \lambda\,(x_1 - x_3) - y_1 \pmod p$$

$$\lambda \equiv \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \pmod p, P \neq Q \\ \dfrac{3{x_1}^2 + a}{2y_1} \pmod p, P = Q \end{cases}$$

Point doubling operation is defined as repeated addition, for example, 3P = P + P + P.

3.3 Weil Pairing

The definition of Weil pairing: Suppose E (F_q) is an elliptic curve defined in a finite field Fq, $P \in E(F_q)$ and $Q \in E(F_q)$, there is a function that satisfies $\mathrm{div}(f_Q) = nD_Q$, then the definition of P and Q's Weil pairing is Eq. (2). Here, $\mathrm{div}(f_Q)$ is the divisor of function f_Q.

$$\hat{e}(P, Q) = \frac{f_P(D_Q)}{f_Q(D_P)} \tag{2}$$

3.4 Bilinear Maps

The bilinear map is $e : G_0 \times G_0 \rightarrow G_1$ implemented via the Weil pairing or Tate pairing. Suppose that k is the secure parameter, and prime order p is k-bits long. Let G_0 and G_1 be two multiplicative cyclic groups of prime order p, and let g be a generator of G_0.

A map $e : G_0 \times G_0 \rightarrow G_1$ is an admissible bilinear map if it satisfies the three following properties:

- Computable. $\forall P, Q \in G_0$, there exists an efficient algorithm to compute $e(P, Q)$;
- Bilinearity. For all $a, b \in \mathbb{Z}_p$, we have $e(g^a, g^b) = \mathrm{e}(g, g)^{ab}$.
- Non-degeneracy. The map is non-degenerate, it must be $\mathrm{e}(g, g) \neq 1$.

3.5 Shamir's Secret Share

Secret distribution: first, the secret sharer needs to construct a $t-1^{\text{th}}$ polynomial, assuming that the secret value to be shared is s. The polynomial $f(x) = s + a_1x + a_2x^2 + \cdots a_nx^{n-1}$ is constructed by randomly selecting $t-1$ elements $a_1, a_2, \ldots a_{t-1}$, and then the secret sharer calculates the secret share $s_i = f(i) \pmod p), \mathrm{i} = 1, 2, \ldots n$, and sends it to the n participants. If a user has t secret shares at least, then he can calculate the shared secret value s.

Secret reconstruction: Assume that $D_n = \{0, 1, \ldots, n-1\}$, we define the Lagrange coefficient $p_k(x)$ in Eq. (3):

$$p_k(x) = \prod_{i \in B_k} \frac{x - x_i}{x_k - x_i} \tag{3}$$

Here, $B_k = \{i | i \neq k, i \in D_n\}$, $p_k(x)$ is a polynomial of $n-1$ degree, which is satisfied $\forall m \in B_k, p_k(x_m) = 0$ and $p_k(x_k) = 1$. At last, we can get the Lagrange interpolation polynomial formula (5) and the expansion of Lagrange interpolation polynomial Eq. (4):

$$f(x) = \sum_{j=0}^{n-1} y_j p_j(x) \tag{4}$$

$$f_n(x) = y_i * \frac{(x - x_2)(x - x_3) \ldots (x - x_n)}{(x_1 - x_2)(x_1 - x_3) \ldots (x_1 - x_{n+1})} \tag{5}$$

Using Eq. (5), the secret value can be calculated, here $s = f(0)$.

4 Access Control Policy and Model

The access policy, named access structure, is the core of ABE, which is exactly a set of data elements authorized to access. In pieces of ABE literature [18–22], it was introduced. To summarize, access structure is mainly divided into the following categories.

4.1 Simple Attribute Set

Assume that S is the attribute set and the threshold value is 2, if someone owns the set S_1, if $S_1 \cap S >=$ threshold, then the person can access this document. For example, $S = \{$Computer, Professor, High-eve$\}$, threshold is 2. $S_1 = \{$Computer, Professor$\}$, $S_1 \cap S = |\{$Computer, Professor$\}| = 2$. This value is satisfied the threshold, so the person can decrypt the document. But in practical application, this solution based on preset thresholds is not universal.

4.2 Bool Algebra

The access control policy is expressed by Boolean algebra, policies are AND-OR relationships between these attributes. A policy is a logical expression consisting of attributes and their relationships. For example, $S =$ ((A and B) and (C or D or E) and (F and G), $S_1 = \{$A, B, D, F, G$\}$. S_1 meets the conditions of access control policy S, so the person with attribute S_1 has the right to access the document.

4.3 Access Tree

In this model, access control policy is represented by an access tree. Therefore, an access tree implies an authorization set.

(1) Access tree = threshold nodes + leaf nodes.

An access tree is used to represent an authorization set. The leaf nodes are the attribute values, and the non-leaf nodes are the threshold nodes. The user needs to meet the minimum value of this threshold. For example, if the threshold is (3, 2) and the node has 3 children, the user needs to satisfy at least 2 children. For example 3: According to the access tree in Fig. 1. The attribute set $S_1 = \{$A, C, D, I, J$\}$, it satisfies the conditions of the access tree, so this user with this set of attributes meets the access conditions.

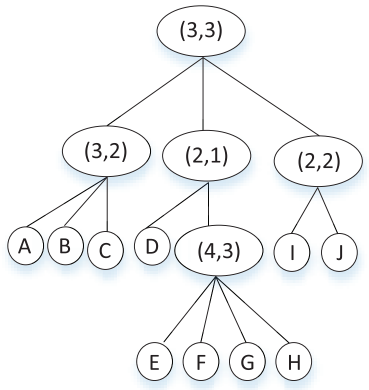

Figure 1: Access tree with threshold node

(2) Linear Secret Sharing Scheme

Linear Secret Sharing Scheme (LSSS) is a rule for transforming the access threshold tree, and the access tree can be transformed into a matrix. The conversion method is for each non-leaf node starting from the root node, the nodes are numbered by top-down hierarchical traversal, in order of 0, 1, 2...; For each leaf node from left to right, top to bottom, numbered from

−1, in order of −1, −2 . . . ; Each row of the matrix is used to store the threshold information and child node numbers of non-leaf nodes. The non-leaf node is stored sequentially in the top-down hierarchical traversal order. To satisfy the neatness of the matrix, insufficient information is filled with 0 to supplement.

For example 4: take Fig. 1 as an example. In this tree, it has 5 non-leaf nodes, (3, 3) is number 0, (3, 2) is number 1, . . . (4, 3) is number 4, and it has 11 leaf nodes, their numbers are A (−1), B (−2), C (−3) . . . J(−10). The corresponding LSSS matrix of Fig. 1 is below. According to the conversion method, the first row represents the root node: its threshold information (3, 3) and its children's numbers: 1, 2, 3, and so on. The second row is the non-leaf node (3, 2), and its children's numbers are −1, −2, −3.

$$\begin{bmatrix} 3 & 3 & 1 & 2 & 3 & 0 \\ 3 & 2 & -1 & -2 & -3 & 0 \\ 2 & 1 & -4 & 4 & 0 & 0 \\ 3 & 2 & -9 & -10 & 0 & 0 \\ 4 & 3 & -5 & -6 & -7 & -8 \end{bmatrix}$$

(3) Access tree = threshold nodes + leaf nodes + polynomials

Neither the above access tree nor the access tree represented by the LSSS matrix is resistant to user collusion attacks. Here, the introduction of random polynomials solves the collusion attack problem. Shamir's secret sharing cannot solve the collusion attack problem, so it must resort to bilinear mapping and cryptography.

For each threshold node, we construct a random polynomial, use a secret sharing scheme to distribute the encryption key layer by layer, and finally go to the leaf node. When accessing, as long as the attribute set makes the root node satisfy the conditions, it can be decrypted. Here, the number of the random polynomial is determined by t in (s, t). For example, for a node with a threshold (3, 2), the number of randomly assigned polynomial terms is 2. For example: Assume the encryption key is 5. For a user, according to the access tree in Fig. 2.

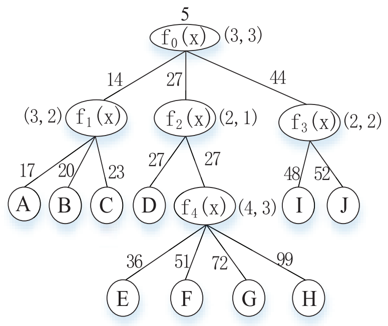

Figure 2: Access tree with threshold and polynomial

Four polynomials *f_0, f_1, f_3 and f_4* are generated at random. Assume that the corresponding thresholds and polynomials are shown in Table 1.

Table 1: Threshold, polynomial and secret shares

Threshold nodes	The random polynomial of the node	The secret shares of node
(3, 3)	$f_0(x) = 5 + 7x + 2x^2$	$f_0(1) = 14\, f_0(2) = 27\, f_0(3) = 44$
(3, 2)	$f_1(x) = f_0(1) + 3x = 14 + 3x$	$f_1(1) = 17\, f_1(2) = 20\, f_1(3) = 23$
(2, 1)	$f_2(x) = f_0(2) = 27$	$f_2(1) = 27\, f_2(2) = 27$
(2, 2)	$f_3(x) = f_0(3) + 4x = 44 + 4x$	$f_3(1) = 48\, f_3(2) = 52$
(4, 3)	$f_4(x) = f_2(2) + 6x + 3x^2 = 27 + 6x + 3x^2$	$f_4(1) = 36\, f_4(2) = 51\, f_4(3) = 72$ $f_4(4) = 99$

For different users, the root encryption key is the same key (the root secret value is 5), but the polynomial is random. For example, in Table 1, the user's random polynomial is $f_0(x) = 5 + 7x + 2x^2$, and another user has a different random polynomial $f_0(x) = 5+3x+4x^2$. Although they have the same root secret value, they each have different polynomials, and naturally, their secret share is distributed differently from the top to down. So, the different user has different private key components. Since the private key components are bound to random polynomials, users cannot implement collusion attacks.

4.4 *Selective-Set Model for ABE*

Init. Define the adversary A and the challenger B. A declares the attribute set S.

Phase 1. A sends the queries for many access structures A_j, where $S \notin A_j$ for all j.

Challenge. A submits two messages M_0 and M_1 with equal length. B flips a random coin b, and encrypts M_b with S. The ciphertext is passed to A.

Phase 2. Phase 1 is repeated.

Guess. A outputs a guess b_0 of b. The advantage of A is defined as $P_r\,[b_0 = b] - 1/2$.

5 ABE Scheme with Message Hierarchy

In practice, the message has different levels of security, and users with different attributes are expected to have different access rights. Here, an ABE scheme with a message security hierarchy was proposed. A message is divided into many blocks according to the security level, and the users with different attribute set keys can decrypt the part of the message with different security levels by access policies.

5.1 *Access Structure*

In this part, a new access structure based on a threshold tree and attribute hierarchy was proposed. The threshold tree is similar to the Huffman tree, and the difference is that it allows each node to have many children greater than 2, and each threshold node saves the corresponding key of encrypting every message block. The higher the level of the threshold node in the tree, the higher the security level of the message block encrypted with it. The structure of child nodes from left to right is the threshold node with message level n − 1, attribute 1, attribute 2, … The detailed access tree is shown in Fig. 3.

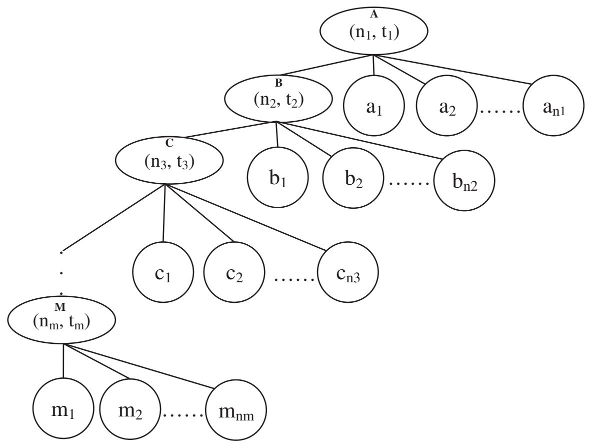

Figure 3: Threshold tree structure for multiple security levels

Similar to the access tree with threshold node and polynomial, this kind of access structure is also resistant to collusion attacks by assigning different random polynomials to users with the help of thresholds and polynomials. To achieve this access with hierarchy, it is necessary to sort the attributes by weight. According to Fig. 3, the weight set is shown in Table 2. The higher the weight value of the attribute set, the lower its privilege.

Table 2: The attribute set and their weights

The weight	Threshold node	Attribute set	Attribute group
1	(n_1, t_1)	$a_1, a_2, \ldots a_{n1}$	U_1
2	(n_2, t_2)	$b_1, b_2, \ldots b_{n2}$	U_2
...	...	...	...
$m-1$	(n_{m-1}, t_{m-1})	...	U_{m-1}
m	(n_m, t_m)	$m_1, m_2, \ldots m_{nm}$	U_m

5.2 The Design and Construction

In our construction, the architecture of the hierarchy ABE is shown in Fig. 4.

- Data Owner (DO): The data owner is the one who stores and shares the data in the ciphertext in the cloud server environment. It is responsible for defining the access structure, encrypting, and uploading the generated ciphertext to the cloud server.
- Key Generating Center (KGC): It is a fully trusted entity and can provide the registration of users. He can also perform a series of initialization operations, generate the secret key and save critical information.
- Cloud Server: It is a not fully trusted entity that can only perform the tasks assigned to him and return the corresponding results. It has computing and storage resources and can be responsible for providing partial decryption for the user.
- Data Users (DU): The users are the people who access the cloud server to download and decrypt the interested ciphertext from the cloud server.

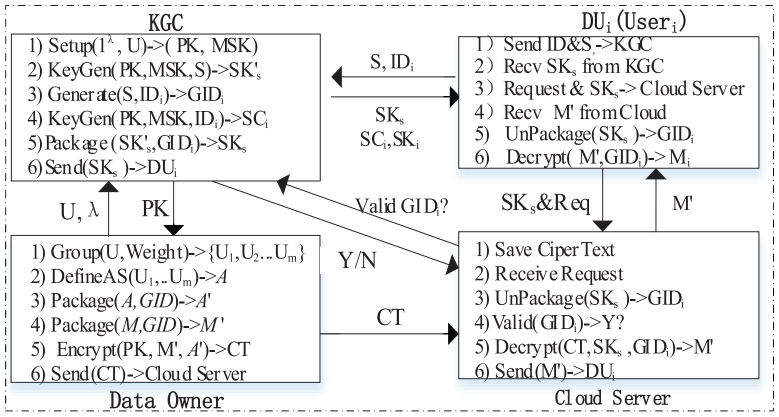

Figure 4: The architecture of the CP-ABE and data flow

5.3 *Security Model*

(1) DefineAS (U)

For the data owners, the important task is to define an access structure that explicitly shows the attributes available to users who have access rights. More specifically, first, associating the specified weight to an attribute based on its importance. Second, dividing the attribute set U into several attribute groups U_1, U_2 ... and U_m according to their weights, the result shows in Table 1. Finally, completing the creation of an access structure A with threshold and multiple security levels. The detailed access tree is shown in Fig. 3.

(2) $Setup(1^{\lambda}, U)$

For the trusted KGC, first, enter security parameter λ and attribute set U, the set U is related to the attribute groups U_1, U_2 ... and access structure. The setup algorithm will choose a bilinear group G_1 of prime order p with generator g. The size of the bilinear group G_0 determines the security parameter λ. In addition, let $e : G_0 \times G_0 \to G_1$ denote the bilinear map, it has the properties of bilinearity and non-degeneracy, as described in Section 3.

In this algorithm, it chooses α, β at random, $\alpha, \beta \in Z_p$. It outputs the Public Key (PK) and a Master Secret Key (MSK). Here, PK is public, and MSK is kept secret, which is used when productizing the user attribute group key.

$Setup(1^{\lambda}, U) \to PK, MSK\ PK : G_0, g, e(g, g),^{\alpha} g^{\beta}\ MSK : g^{\alpha}, \beta$

(3) KeyGen (PK, MSK, S)

In this phase, the system will generate the user's decryption key SK_s. This algorithm takes as input the public parameters PK, the master secret key MSK and a set of attributes S. It outputs the user's secret key SK_s. The steps of key generation and distribution are as follows:

First, according to the user's attribute set S, the system provides its group GID_i based on its attribute weight. And then choose a polynomial for each threshold node t_x in the tree, sending the degree d_x of the polynomial q_x to be one less than the threshold value t_x of that node, that is, $d_x = t_x - 1$. Then, start with the root node $R(n_1, t_1)$, choose a random value y, and set $q_R(0) = y$. It chooses d_R other nodes of the polynomial q_R randomly to define it completely. For node x, it sets $q_x(0) = q_{parent(x)}$ (index(x)) and chooses d_x other nodes randomly to completely define q_x. Next, choose a random value r for the user DU$_i$ with attribute set S, and then choose a random r$_j$ for each valid attribute of this user. At last it computes the key set SK'_S as Eq. (6):

$$SK'_S = \left(D = g^{\frac{\alpha}{\beta}} g^{\frac{r}{\beta}}, \forall j \in S, D_j = g^r \cdot \mathrm{H}\,(att\,(j))^{r_j}, D'_j = g^{r_j}\right) \tag{6}$$

In this part, and then, designing an embedding algorithm. It takes SK'_S and GID$_i$ as the input parameters, and the final secret keys SK_S are formed.

$Package(SK'_S, \mathrm{GID}_i) \rightarrow SK_S$

Finally, the system will distribute the key SK$_s$ to the user DU$_i$.

(4) Encrypt (PK, M, A)

This algorithm takes the PK (public parameters), M (shared message), and A (an access structure) as the input parameters, and it outputs the ciphertext CT. Encryption with PK and M proceeds as follows. First, to prevent information leakage in access structure A, the group keys GID are integrated into the access structure. In this way, it not only achieves a hidden access structure but also carries the group key to complete the secondary decryption by the user. Then, a random value t is chosen, and let attribute set U be the set of leaf nodes in source access structure A. The ciphertext constructed by giving the tree access structure *A'* is published below Eq. (7):

$Package(A, GID) \rightarrow A'$ $Package(M, GID) \rightarrow M'$

$$CT = \left(A', \widetilde{C} = M'e\,(g, g)^{\alpha t}, C = g^{\beta t}, \forall x \in U, C_x = g^{q_{x(0)}}, C'_x = \mathrm{H}\,(\mathrm{att}\,(x))^{q_{x(0)}}\right) \tag{7}$$

(5) Decrypt (PK, CT, SK$_s$)

The decryption algorithm consists of two sub-algorithms, one runs in the cloud server, and it completes the partial decryption of the ciphertext, and the other runs in the DU, and decrypts the whole by the users. It takes as input the public parameters PK, a secret key SK$_s$ and a ciphertext CT. It outputs the message M'. If the attribute set S satisfies the access structure A, then Decrypt (PK, CT, SKs)→M'. Otherwise, with overwhelming probability, Decrypt (.) outputs a random message. In this phase: the decryption user gets the corresponding ciphertext, and the user decrypts it with his own key SK$_s$ to get the plaintext message M.

For the overall encryption process, we first choose an arbitrary d-element subset. In Fig. 3, assume that: the threshold of node A is $n_1 = 5$ and $t_1 = 3$, that is A (5, 3). We assume that the user's attribute key set contains the keys for attribute nodes a$_1$, a$_2$ and a$_4$, which means that we can compute: $e(g, g)^{rq_{a1}(0)}$, $e(g, g)^{rq_{a2}(0)}$ and $e(g, g)^{rq_{a4}(0)}$.

Further, based on the secret sharing and the bilinear map operation, we know that Eqs. (8)–(10):

$$e(g, g)^{rq_{a1}(0)} = e(g, g)^{rq_A(1)} \tag{8}$$

$$e(g, g)^{rq_{a2}(0)} = e(g, g)^{rq_A(2)} \tag{9}$$

$$e(g, g)^{rq_{a4}(0)} = e(g, g)^{rq_A(4)} \tag{10}$$

Suppose Lagrange coefficients of node A at x = 1 and x = 2 are $\Delta_1(x)$, $\Delta_2(x)$ and $\Delta_4(x)$, then the Eq. (11) is below:

$$\begin{aligned}
\Delta_1(x) &= \frac{x-2}{1-2}\cdot\frac{x-4}{1-4}\\
\Delta_2(x) &= \frac{x-1}{2-1}\cdot\frac{x-4}{2-4}\\
\Delta_4(x) &= \frac{x-1}{4-1}\cdot\frac{x-2}{4-2}
\end{aligned} \tag{11}$$

Based on the Lagrange interpolation polynomial (5), we can get Eq. (12):

$$q_A(x) = q_A(1)\Delta_1(x) + q_A(2)\Delta_2(x) + q_A(4)\Delta_4(x) \tag{12}$$

Then put x = 0 into the above equation, and the result is Eq. (13):

$$q_A(0) = q_A(1)\Delta_1(0) + q_A(2)\Delta_2(0) + q_A(4)\Delta_4(0) \tag{13}$$

Therefore, Eq. (14) is

$$\begin{aligned}
&(e(g,g)^{rq_A(1)})^{\Delta_1(0)} \cdot (e(g,g)^{rq_A(2)})^{\Delta_2(0)} \cdot (e(g,g)^{rq_A(4)})^{\Delta_4(0)}\\
&= e(g,g)^{rq_A(1)\Delta_1(0)+rq_A(2)\Delta_2(0)+rq_A(4)\Delta_4(0)}\\
&= e(g,g)^{rq_A(0)}\\
&= e(g,g)^{s}
\end{aligned} \tag{14}$$

Next step:

$$UnPackage(A') \rightarrow A, GID$$

$$UnPackage(SK_S) \rightarrow \mathrm{GID}_i$$

Separating the GID_i from the user group key SK_S, if $GID_i \notin GID$, then the decryption is terminated, otherwise using the secret value y of root node and access structure A, take C, D and A into the Eq. (15), we can get *M'* according to Eq. (16).

$$\begin{aligned}
&\widetilde{C}/\left(e(C,D)\right/\left(\prod_{i\in S_x}\left(e(C_x,D_i)/e\left(C'_x,D'_i\right)\right)^{\Delta_{i,s}(0)}\right)\\
&= \widetilde{C}/\left(e(C,D)\right/\left(\prod_{i\in S_x}\left(e\left(g^{q_x(0)}, g^r\cdot H(att(x))^{r_i}\right)/e\left(H(att(x))^{q_x(0)}, g^{r_i}\right)\right)^{\Delta_{i,s}(0)}\right)\\
&= \widetilde{C}/\left(e(C,D)\right/\left(\prod_{i\in S_x}\left(e\left(g^{q_x(0)}, g^r\right)\cdot e\left(g^{q_x(0)}, H(att(x))^{r_i}\right)/e\left(H(att(x))^{q_x(0)}, g^{r_i}\right)\right)^{\Delta_{i,s}(0)}\right)\\
&= \widetilde{C}/\left(e(C,D)\right/\left(\prod_{i\in S_x}\left(e\left(g^{q_x(0)}, g^r\right)\cdot e\left(g^{q_x(0)}, H(att(x))^{r_i}\right)/e\left(g^{q_x(0)}, H(att(x))^{r_i}\right)\right)^{\Delta_{i,s}(0)}\right)\\
&= \widetilde{C}/\left(e(C,D)\right/\left(\prod_{i\in S_x} e(g,g)^{\mathrm{r}\cdot q_x(0)\cdot\Delta_{i,s}(0)}\right)\\
&= \left(\widetilde{C}/\left(e(C,D)\right)\cdot e(g,g)^{rq_x(0)}\right) = \left(\widetilde{C}/\left(e(C,D)\right)\cdot e(g,g)^{rs}\right)\\
&= \frac{\widetilde{C}\cdot e(g,g)^{rs}}{e(C,D)}
\end{aligned} \tag{15}$$

$$\frac{\widetilde{C}\cdot e\left(g,g\right)^{rs}}{e\left(C,D\right)}$$
$$=\frac{M'e(g,g)^{\alpha s}\cdot e(g,g)^{rs}}{e\left(g^{\beta s},g^{\frac{\alpha}{\beta}}g^{\frac{r}{\beta}}\right)}$$
$$=\frac{M'e(g,g)^{\alpha s}\cdot e(g,g)^{rs}}{e\left(g^{\beta s},g^{\frac{\alpha+r}{\beta}}\right)} \tag{16}$$
$$=\frac{M'e(g,g)^{\alpha s}\cdot e(g,g)^{rs}}{e\left(g,g\right)^{\beta s\cdot\frac{\alpha+r}{\beta}}}$$
$$=\frac{M'e(g,g)^{\alpha s}\cdot e(g,g)^{rs}}{e(g,g)^{\alpha s}\cdot e(g,g)^{rs}}$$
$$=M'$$

At last, Message M' is sent to the user DU_i, by his group secret key SK_S, the unpacked and decryption function is used, and he can get the corresponding plaintext message M_i.

$UnPackage(SK_S)\rightarrow \mathrm{GID}_i$
$Decrypt(M',\mathrm{GID}_i)\rightarrow M_i$

In the tradition ABE decryption algorithm, using the secret value y of the root node and access structure A, take C, D and A to decrypt the ciphertext, it needs to distinguish different nodes. If node x is a leaf node, then we can decrypt the nodes by Eq. (15), if x is a non-leaf node, it calls a recursive decryption algorithm to proceed with the process. For all nodes z that are children of x, it calls DecrptNode(CT, SK, z) and stores the output as F_z.

5.4 Proof of Security

In this section, we prove the security of our scheme. The security game is between a simulator W and an adversary P, the game proceeds as follows:

Init. W runs the algorithm and P chooses the attribute set S. This challenge is divided into two phases: one is the acquisition of legitimate GID_s, and the other is the access structure. Assume that the system provides the adversary the attribute group GID_s based on its attribute weight, and satisfied $\mathrm{P_r}[GID_s\in GID]=1/2$.

Setup. The simulator assigns the public key parameters as follows. It chooses random $\alpha,\beta\in Z_p$, which we associate with the integers from 0 to $p-1$, (p is the prime order of the bilinear group G_0 with generator g), and the public parameter $Y=e(g,g)^a$, g^β are sent to the adversary. From the view P all parameters are chosen at random as in the construction.

When the adversary P calls for the evaluation of $H\ (attr_i)$. (Hash on any attribute string $attr_i$), a new random value $t_i\in Z_p$ is chosen, the simulator uses g^{t_i} to response the $H\ (attri)$.

Phase 1. When the adversary makes its j^{th} key generation query for the set S_j of attributes, a new random value $\mathrm{r}^{(j)}$ is chosen from Z_p and for every $\mathrm{i}<S_{j,}$ new random values $r_i^{(j)}$ are chosen from Z_p. The simulator computes: $D=g^{\frac{\alpha}{\beta}}g^{\frac{r(j)}{\beta}}$ and for each $i<S_j$, we have $D_i=g^{r^{(j)}}.g^{t_i.r_i^{(j)}}$ and $D_i'=g^{r_i^{(j)}}$, these values are passed onto the adversary.

Suppose P makes a request for an access structure T where T(S) = 0. To generate the secret key, W needs to assign a polynomial Q_x of degree d_x for every node in T. We define two function Satisfy and Unsatisfied. Satisfy (T_x, S, λ_x) sets up the polynomials for the nodes of an access sub-tree with satisfied root node, that is, $T_x(S) = 1$; Unsatisfy (T_x, S, g^{λ_x}) sets up the polynomials for the nodes of an access tree with unsatisfied root node, that is, $T_x(S) = 0$.

For each child node x' of x, the algorithm calls: Satisfy ($T_{x'}$, S, q_x(index(x'))), if x' is a satisfied node. Unsatify (T_x, S, $g^{q_x(\text{index}(x'))}$), if x' is not a satisfied node. To give keys for access structure T, the simulator runs Unsatify (T, S, P) to define a polynomial q_x for each node x of T. For each leaf node x of T, we know q_x completely if x is satisfied, if x is not satisfied, then at least $g^{q_x(0)}$ is known.

Simulator now defines the final polynomial $Q_x(\cdot) = bq_x(\cdot)$ for each node x of T . Notice that this sets y = $Q_s(0)$ = ab. The key corresponding to each leaf node is given using its polynomial as follows. Let i = att(x).

If $i \in S$, $D_x = g^{\frac{Q_x(0)}{t_i}} = g^{\frac{bq_x(0)}{t_i}}$, and if $i \notin S$, $D_x = g^{\frac{Q_x(0)}{t_i}} = g^{\frac{bq_x(0)}{b\beta_i}} = g^{\frac{q_x(0)}{\beta_i}}$.

Here, constructing the private keys for the access structure T and distributing the private keys for T are identical to that in the original scheme.

Challenge. The adversary P submits two equal length messages m_1, m_0 to the simulator W. The simulator flips a fair binary coin v, and encrypts m_v. The ciphertext is passed to P.

Phase 2. The simulator repeated it as in Phase1.

Guess. Assume that the adversary's advantage in this situation is ε. A submits a guess v' of v. If v' = v the simulator outputs u' = 0, otherwise output µ' = 1. The probability analysis under different circumstances is shown in Table 3.

Table 3: The probability analysis

The probability in different cases	The description
$P_r[v \neq v' \mid u = 1] = \frac{1}{2}$	The adversary gains nothing about v where u = 1
$P_r[u' = u \mid u = 1] = \frac{1}{2}$	The simulator guesses u' = 1 when $v \neq v'$
$P_r[v = v' \mid u = 0] = \frac{1}{2} + \varepsilon$	The adversary sees an encryption of m_v where u = 0
$P_r[u' = u \mid u = 0] = \frac{1}{2} + \varepsilon$	The simulator guesses u' = 0 when $v = v'$

Synthesize the above conditions, in the second phase, the advantage of the simulator is $\left(\frac{1}{2}P_r[\text{u}' = u \mid u = 0] + \frac{1}{2}P_r\left[\text{u}' = u \mid u = 1\right] - \frac{1}{2}\right) = \frac{\varepsilon}{2}$. So in two-stage process, the overall advantage of the simulator in the game is $\frac{1}{2} * \left(\frac{1}{2}P_r[\text{u}' = u \mid u = 0] + \frac{1}{2}P_r\left[\text{u}' = u \mid u = 1\right] - \frac{1}{2}\right) = \frac{1}{2} * \frac{\varepsilon}{2} = \frac{\varepsilon}{4}$.

5.5 Experimental Results and Discussion

To evaluate the performance, this paper implements it based on CP-ABE, and selects different security levels to test. The test was conducted on a computer with Inter (R) Core (TM) i5-6500 CPU

3.20 GHz and 16 GB of RAM. We use IntelliJ IDEA as the integrated development environment. Java is used as the programming language, and JPBC libraries are used to implement the operation (https://crypto.stanford.edu/pbc/). PBC library classifies the pairs into type A, B∼G seven categories, and we use an elliptic curve group constructed on the curve $y^2 = x^3 + x$ based on type A.

Here, the safety coefficient is 80, 160, and 320 bits, respectively, and the prime order of base point is 256, 512, and 512 bits, respectively, so there are three groups to test the performance: type A-80-256, type A-160-512, type A-320-512. The hash function is implemented using SHA-256. The number of given attributes (according to the attribute weight, the message hierarchy was set from 1 to 5) increases from 1 to 10. The testing went through 20 rounds, and the average value was as the final time consumption.

(1) **Key generation time at different security levels**

In this experiment, we set the access policy as below: set the message hierarchy as 5, in every level, its number of attributes changed from 1 to 10, and all the threshold is AND gate.

① Testing the setup time in different security conditions. From the test result we know that when we select type A, the safety coefficient is 80, and the prime order is 256 bits, its setup time is 24.442 ms, when we select type A-160-512, the setup time is 51.754 ms, and when we select type A-320-512, the setup time is 83.008 ms. With the increase in security level, its setup time is increasing.

② Testing the key generated time in different conditions. We set an access policy with 5 message hierarchies, and each message hierarchy has the same number of attributes. When each message level has 1 attribute node, its key generation time is 20.362 ms in type A-80-256, and when each message level has 2 attribute nodes, its key generation time is 27.405. With the increase of attribute number in every message hierarchy, the key generation time is gradually linear increasing. In the same way, at different security levels, the key generation time is different, the detailed information is shown in Table 4.

Table 4: The key generation time at different security levels (Unit: ms)

Type A-80-256 Round: 10 AttrNum: 1∼10 Message Hierarchy: 5 Setup Time: 24.442										
AttrNum:	1	2	3	4	5	6	7	8	9	10
KeyGen:	20.362	27.405	34.069	38.865	45.935	59.181	61.702	70.614	74.719	88.098
Type A-160-512 Round: 10 AttrNum: 1∼10 Message Hierarchy: 5 Setup Time: 51.754										
KeyGen:	84.851	114.612	152.53	183.073	211.231	235.377	267.201	298.64	331.617	357.655
Type A-320-512 Round: 10 AttrNum: 1∼10 Message Hierarchy: 5 Setup Time: 83.008										
KeyGen:	171.155	241.394	307.382	369.148	418.004	488.687	557.75	607.175	674.613	750.113

(2) **Encryption time at different security levels**

In this part, we set the encryption time under different conditions. Similar to test (1), the setup time has not changed. We set an access policy with 5 message hierarchies, and each message hierarchy has the same number of attributes. When each message level has 1 attribute node, its encryption time is 43.567 ms in A-80-256, and when each message level has 2 attribute nodes, its key generation time is 49.414 ms. With the increase of attribute number, the encryption time is gradually linear increasing.

In the same way, at different security levels, the encryption time is different, the information is shown in Table 5.

Table 5: The encryption time at different security levels (Unit: ms)

Type A-80-256 Round: 10 AttrNum: 1∼10 Message Hierarchy: 5 Setup Time: 24.442										
AttrNum:	1	2	3	4	5	6	7	8	9	10
Encryp:	43.567	49.414	58.535	67.676	82.834	88.687	99.37	113.079	121.547	131.383
Type A-160-512 Round: 10 AttrNum: 1∼10 Message Hierarchy: 5 Setup Time: 51.754										
Encryp:	131.594	185.984	233.809	286.286	342.525	390.447	441.926	497.866	547.624	602.87
Type A-320-512 Round: 10 AttrNum: 1∼10 Message Hierarchy: 5 Setup Time: 83.008										
Encryp:	260.584	356.987	462.901	573.861	675.184	818.616	897.834	997.635	1102.793	1209.286

(3) **Decryption time cost at different security levels**

In this part, we set the decryption time in different conditions. Similar to test (1), the setup time hasn't changed. We set an access policy with 5 message hierarchies, and each message hierarchy has the same number of attributes. When each message level has 1 attribute node, its decryption time is 39.388 ms in A-80-256, and when each message level has 2 attribute nodes, its key generation time is 47.935 ms. With the increase of attribute number, the decryption time is gradually linear increasing. In the same way, at different security levels, the decryption time is different, the detailed information is shown in Table 6.

Table 6: The decryption time cost at different security levels

Type A-80-256 ROUND = 10 AttrNum: 1∼10 Setup Time: 24.442										
AttrNum:	1	2	3	4	5	6	7	8	9	10
Decryp:	39.388	47.935	65.861	75.386	85.688	94.269	103.679	114.23	125.317	135.518
Type A-160-512 ROUND = 10 AttrNum: 1∼10 Setup Time: 51.754										
Decryp:	126.711	184.562	234.718	285.666	337.22	387.277	439.88	495.73	553.984	601.351
Type A-320-512 ROUND = 10 AttrNum: 1∼10 Setup Time: 83.008										
Decryp:	233.498	343.713	446.327	548.966	652.82	740.344	840.409	946.32	1064.088	1155.175

(4) **The discussion of the solution**

In this part, we discuss the dataset, number of users, key size, and computation time. In the ABE scenario, the size of the data set is associated with the shared message, which mainly comes from the data owner, and the number of users is linked with the attribute set. In our scheme, the access rights are refined so that users have different access perspectives to the same shared message. Regarding ciphertext size and private key size, let G_0 and G_1 be two multiplicative cyclic groups of prime order p. Let g be a generator of G_0, Z_p be the group $\{0, 1, \ldots, p-1\}$, and the attributes can be denoted as $\{A_{U1}, A_{U2}, \ldots A_{Um}\}$. Assume that L_* is the bit-length of the element in $*$, then the size of PK is

$3L_{G0} + L_{G1}$, the size of MSK is $L_{zp} + L_{G0}$, and the length of the user key SK_{Ui} is $|A_{Ui}| * L_{G0} + L_{Gid}$. The number of group elements grows linearly with the number of attributes associated with her identity. In terms of time complexity, attributes were grouped and weighted according to their importance of attributes, and the access structure based on a threshold tree was constructed according to attribute weight. Suppose there are n threshold nodes in the access structure tree, and the average operation time to compute a threshold node is t, then in the previous ABE, the time to decrypt the root node is $n * t$, while in our scheme, the decryption time is t. So there is a significant improvement in the process of getting the decryption key.

6 Conclusion

At present, CP-ABE is considered to be the ideal high-security data protection solution with protection against collusion or malicious attacks in a shared environment. But in practical applications, the shared message needs different access perspectives for users with different permissions, for this scenario, we proposed a lightweight new ABE scheme.

First, we presented our construction of a message security hierarchy CP-ABE scheme that uses attribute weight as the access rank of the message, which makes the users get different levels of messages. And then, the two-step decryption method is used in this scheme, which reduces the computational and storage burden of the users by leveraging the computing power of the cloud. In addition, in this scheme, the introduction of the group ID and user ID makes it possible for the audit to track malicious users. Finally, the security proof and the experiment result show that this scheme has good performance. ABE schemes in cloud environments have broad prospects for development [23,24], therefore, we will continue to study how to better play the role of ABE in different application scenarios.

Funding Statement: This work was funded by the Funding of Nanjing Institute of Technology No. JXGG2021017, the National Natural Science Foundation of China No. 61701221.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. of CRYPTO 84 on Advances in Cryptology*, Berlin, Germany, vol. 196, pp. 47–53, 1985.
[2] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.
[3] A. Sahai and B. Waters, "Fuzzy identity based encryption," in *Advances in Cryptology EUROCRYPT*, Heidelberg, Springer, vol. 3494, pp. 457–473, 2005.
[4] J. Bethencourt, A. Sahai and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symp. on Security and Privacy*, Berkeley, USA, pp. 321–334, 2007.
[5] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Int. Workshop on Public Key Cryptography (PKC)*, Berlin, Germany, pp. 53–70, 2011.
[6] V. Goya, O. Pandey, A. Sahai and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of the 13th ACM Conf. on Computer and Communications Security*, Alexandria, VA, USA, pp. 89–97, 2006.
[7] D. Boneh, A. Sahai and B. Waters, "Functional encryption: A new vision for public key cryptography," *Communications of the ACM*, vol. 55, no. 11, pp. 56–64, 2012.

[8] S. Garg, C. Gentry, S. Halevi and M. Zhandry, "Functional encryption without obfuscation," in *Proc. of the Theory of Cryptography Conf. (TCC)*, Berlin, Germany, pp. 1–40, 2016.

[9] Y. Deng, C. Tang, G. Song and Y. Wen, "New cryptography primitive research: Process based encryption," *Journal of Software*, vol. 28, no. 10, pp. 2722−2736, 2017.

[10] Y. Ma, X. Cao, Z. Zheng, H. Guang and B. Wen, "Accountable CP-ABE with public verifiability: How to effectively protect the outsourced data in cloud," *International Journal of Foundations of Computer Science*, vol. 28, no. 6, pp. 705–723, 2018.

[11] S. Banerjee, S. Roy, V. Odelu, A. K. Das and Y. Park, "Multi-authority CP-ABE-based user access control scheme with constant-size key and ciphertext for IoT deployment," *Journal of Information Security and Applications*, vol. 53, 102503, 2020.

[12] S. Wang, J. Zhou and J. Liu, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1265–1277, 2016.

[13] J. Li, N. Chen and Y. Zhang, "Extended file hierarchy access control scheme with attribute based encryption in cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 983–993, 2021.

[14] W. Zhang, Z. Zhang and H. Xiong, "PHAS-HEKR-CP-ABE: Partially policy-hidden CP-ABE with highly efficient key revocation in cloud data sharing system," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 1, pp. 613–627, 2022.

[15] H. Song, F. Yin, X. Han, T. Luo and J. Li, "MPLDS: An integration of CP-ABE and local differential privacy for achieving multiple privacy levels data sharing," *Peer-to Peer Networking and Applications*," vol. 15, no. 1, pp. 369–385, 2022.

[16] N. Chen, J. Li, Y. Zhang and Y. Guo, "Efficient CP-ABE scheme with shared decryption in cloud storage," *IEEE Transactions on Computers*, vol. 71, no. 1, pp. 175–184, 2022.

[17] D. Kumar, M. Kumar and G. Gupta, "An outsourced decryption ABE model using ECC in internet of things," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 29, no. 6, pp. 949–964, 2021.

[18] W. Y. Hwang and I. Y. Lee, "A study on data sharing system using ACP-ABE-SE in a cloud environment," *International Journal of Web and Grid Services*, vol. 17, no. 3, pp. 201–220, 2021.

[19] J. Zhang and H. Gao, "A compact construction for non-monotonic key-policy attribute-based encryption," *International Journal of High Performance Computing and Networking*, vol. 13, no. 3, pp. 321–330, 2019.

[20] E. Affum, X. Zhang, X. Wang and J. B. Ansuura, "Efficient lattice CP-ABE AC scheme supporting reduced-OBDD structure for CCN/NDN," *Symmetry*, vol. 12, no. 1, pp. 166–178, 2020.

[21] Z. Zhang, J. Zhang, Y. Yuan and Z. Li, "An expressive fully policy-hidden ciphertext policy attribute-based encryption scheme with credible verification based on blockchain," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8681–8692, 2022.

[22] S. Y. Tan, K. W. Yeow and S. O. Hwang, "Enhancement of a lightweight attribute-based encryption scheme for the internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6384–6395, 2020.

[23] F. Cheng, S. Ji and C. Lai, "Efficient CP-ABE scheme resistant to key leakage for secure cloud-fog computing," *Journal of Internet Technology*, vol. 23, no. 7, pp. 1461–1471, 2022.

[24] P. K. Premkamal, S. K. Pasupuleti and P. Alphonse, "Dynamic traceable CP-ABE with revocation for outsourced big data in cloud storage," *International Journal of Communication Systems*, vol. 34, no. 2, pp. 1–21, 2021.