2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA)

An ECC-Based Multi-Authority Hierarchical Data CP-ABE Scheme for Industrial IoT

Rurong Chen
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
chenrurong1115@163.com

*Jinnan Zhang
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
zhangjinnan@bupt.edu.cn

Jiehan Qiao
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
qiaojiehan1@163.com

Zhengliang Yuan
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
yzl13515106716@163.com

Qinghua Gong
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
qinghuagong@bupt.edu.cn

Yihan Wang
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
3188776763@qq.com

Xin Yan
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
xyan@bupt.edu.cn

Xueguang Yuan
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
yuanxg@bupt.edu.cn

***Abstract*—The rapid development of the Industrial Internet of Things (IIoT) has raised higher demands for the secure sharing, collaboration, and real-time interaction of massive sensor data in the cloud. Ciphertext-Policy Attribute-Based Encryption (CP-ABE) is considered a potential solution to address this issue. However, existing schemes do not take into account the resource limitations of intelligent sensing terminals and the hierarchical sensitivity of actual industrial data. To address this issue, this paper proposes a multi-authority CP-ABE scheme for hierarchical industrial IoT data based on Elliptic Curve Cryptography (ECC). The scheme eliminates redundant pairing operations in favor of scalar multiplication and decouples the decryption phase through edge computing, utilizing multiple authority centers for attribute and key management. Additionally, it replaces the traditional hierarchical access tree with a global Linear Secret Sharing Scheme (LSSS), embedding blinding factors for multiple files and enabling collaborative encryption of multi-level files by embedding lower-level ciphertext information into higher levels through the concept of hierarchical bundling. This approach effectively reduces the number of attribute matching rounds. Security analysis and performance evaluations demonstrate that the proposed scheme offers significant advantages in resource-constrained Industrial IoT environments.**

Keywords-Industrial Internet of Things; attribute-based encryption; pairing free; hierarchical sharing; multiple authorities; linear secret sharing

I. INTRODUCTION

The Industrial Internet of Things (IIoT) is a technology that connects various industrial devices, sensors, control systems, and other physical equipment to the internet or internal networks, enabling data collection, transmission, analysis, and control. However, due to the enormous number of devices in IIoT networks and the rapid influx of massive, multi-source, hierarchical data, local storage becomes inadequate. Cloud Service Providers (CSPs) offer businesses vast storage capacity and efficient computing services. Nevertheless, when data owners upload their private data to cloud servers for storage, the service providers effectively become the true owners of the data, and users lose control over their private data. This presents a significant challenge in protecting the privacy and security of data stored on cloud platforms.

The advent of bilinear pairings has helped resolve many issues that were previously unsolvable in the field of cryptography. Based on bilinear pairings, Sahai and Waters [1] first proposed the Attribute-Based Encryption (ABE) scheme in 2005. Two year later, Bethencourt et al. [2] proposed the first ciphertext-policy attribute-based encryption scheme. To address the single-point bottleneck and trust issues associated with a central authority, the Multi-Authority Attribute-Based Encryption (MA-ABE) scheme [3] and decentralized MA-ABE schemes [4]-[5] were later proposed. However, the large decryption computation costs hindered the application and development of MA-ABE in resource-constrained devices. M.

2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA) | 979-8-3315-0976-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/AIITA65135.2025.11047796

Gupta et al. [6] introduced an efficient CP-ABE scheme for cloud-based industrial smart vehicles. To reduce the complexity of bilinear pairing operations during decryption, Schemes [7]-[8] propose outsourcing partial encryption and decryption tasks to a third party. Building upon this, the former hides the access policy, while the latter supports attribute updates, making access control more flexible.

However, in practice, the data types in IIoT are diverse and complex, with different types of data requiring different sensitivity levels of access control. Most Ciphertext-Policy Attribute-Based Encryption(CP-ABE) schemes do not consider this hierarchical relationship, requiring data owners to generate multiple access structures to encrypt these files, which inevitably leads to additional computational overhead. To address this, Wang et al. [9] were the first to propose a hierarchical file attribute-based encryption scheme supporting a tree-based access structure to solve the hierarchical file issue. Subsequent research proposed MA-ABE schemes with hidden access policies for hierarchical files [10]-[12], further protecting user privacy. However, these schemes only support tree-based access structures, suffer from low encryption and decryption efficiency, and have poor scalability. Li et al. [13], building on the work in [9], extended its functionality without significantly compromising efficiency, making it more suitable for organizations or companies with large hierarchical structures. Subsequently, He et al. [14] proposed an efficient attribute-based hierarchical data access control scheme supporting the LSSS structure in cloud computing. However, the scheme only supports a single authority center.

All the above schemes involve numerous bilinear pairings and modular exponentiation operations, and the high computational cost severely limits their application in resource-constrained devices in the IIoT. To fundamentally improve algorithm efficiency, Odelu and Das [15] proposed a constant key length CP-ABE scheme based on elliptic curve cryptography, but the scheme only supported AND-gate access structures, limiting its flexibility. Later, Ding et al. [16] proposed a pairing-free CP-ABE scheme supporting LSSS access structures and enabling direct user attribute revocation. However, this scheme, being single-authority, is prone to compromise. Schemes [17]-[21] propose outsourcing decryption computations to a third party. Among them, scheme [19] considers time and location attributes, achieving finer-grained access control. However, none of these schemes take into account the hierarchical characteristics of files in real-world applications.

In our scheme, the redundant bilinear pairings used in traditional CP-ABE are discarded, and the decryption computation is outsourced to edge proxies through edge computing technology, effectively reducing the computational overhead on intelligent sensing terminals. A multi-level Linear Secret Sharing Structure (LSSS) is adopted in place of the traditional hierarchical access tree, leveraging the concept of hierarchical bundling. This approach not only reduces ciphertext storage space but also minimizes the processing of irrelevant information, enabling fast one-time sharing of multiple files. It is more suitable for large-scale distributed network environments such as cloud computing and edge computing.

The remainder of this paper is organized as follows: Section II introduces the relevant theoretical background; Section III presents the system model and detailed scheme design; Section IV provides the security analysis of the scheme; Section V gives the performance analysis and comparison of the scheme; Section VI concludes the paper.

II. PRELIMINARIES

A. Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a public key encryption algorithm based on the Elliptic Curve Discrete Logarithm Problem (ECDLP). An elliptic curve E defined on the finite field $FG(p)$ can be expressed as $y^2 = x^3 + ax + b(mod)p$ and $4a^3 + 27b^2 \neq 0$. Given two points $P, G \in G_E$, where G is the generator elliptic curve group G_E, with prime order r, it is extremely difficult to compute $k \in Z_r$ in polynomial time for the given point $P = kG$.

B. Decisional Diffie–Hellman Assumption

The definition of the Decisional Diffie-Hellman (DDH) assumption on an elliptic curve is described as follows: The challenger selects a cyclic group G_E of prime order r. Let G be a generator of the cyclic group G_E and a, b be randomly chose from Z_r. If the challenger gives the adversary a tuple (G, aG, bG), it must be difficult for the adversary to distinguish a valid element $abG \in G_E$ from a random element $R \in G_E$. The advantage of an algorithm $\mathcal{B}$ in breaking the DDH assumption in G_E is ε if $|\Pr[\mathcal{B}(G, aG, bG, Z = abG) = 0] - \Pr[\mathcal{B}(G, aG, bG, Z = R) = 0]| \geq \varepsilon$.

If the advantage of a polynomial-time algorithm in solving the DDH problem is negligible, then the DDH assumption holds..

C. Linear Secret Sharing Scheme

Let the set of participants be P. If the following conditions are satisfied, then Π is a Linear Secret Sharing Scheme (LSSS) defined over P.

1) A vector over Z_r^* is generated by the secret of each participant.

2) There exists an $l \times n$ sharing matrix A generated by the secret sharing scheme Π, For $i = 1, 2, \ldots l$, the ith row of this matrix, A_i, is identified by a corresponding participants $\rho(i)$, where $\rho(i)$ is a mapping function from $\{1, 2, \ldots l\}$ to P. Given a random column vector $\vec{v} = (s, y_2, y_3, \ldots, y_n)$, where $s \in Z_r$ is the shared secret value, and $y_2, y_3, \ldots, y_n$ are generated by random numbers. $(\mathrm{A} \cdot \vec{v})$ represents the process of splitting the secret value s into l shares according to the sharing scheme Π. $\lambda_i = (\mathrm{A} \cdot \vec{v})_i$ indicates the share attribute corresponding to user $\rho(i)$.

Each linear secret sharing scheme can be reconstructed linearly as defined above, specified as follows.

Assume that there exists an LSSS Π corresponding to the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be defined as any authorized attributes, and at the same time, define $I = \{i : \rho(i) \in S\}$, where $I \subset \{1, \ldots, l\}$. Then there exists a set of constants $\{c_i \in Z_r\}_{\mathrm{i} \in \mathrm{I}}$ in polynomial time such that, if the shares λ_i satisfy the shared

secret value s then: $\sum_{i\in I} c_i \cdot \lambda_i = \sum_{i\in I} c_i \cdot (\mathrm{A}\cdot\vec{v}) = \varepsilon\cdot\vec{v} = s$ and $\varepsilon = (1,0,\dots 0)$. For the unauthorized set, there exists $c_i \cdot \mathrm{A}^T = 0$.

D. Hierarchical Global Access Structure

The hierarchical global access control structure clusters multiple hierarchical access policies into a single access structure P. These access policies have a subordinate relationship, meaning lower levels are contained within higher levels, such that $P_q \subset P_{q-1} \subset \cdots \subset P_1$. In the hierarchical global access structure P, leaf nodes represent attributes, while non-leaf nodes represent "AND" or "OR" gates. The aggregation process is illustrated in Fig. 1.

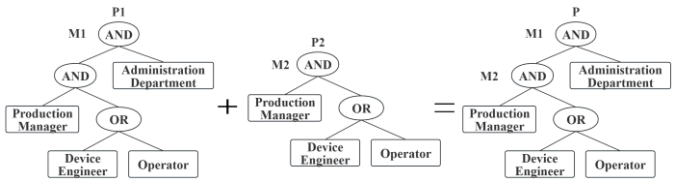

Figure 1. Access Policy Aggregation

Therefore, for access structures with hierarchical relationships, the aggregated access structure P can be converted into a Linear Secret Sharing Structure (A, ρ) based on the matrix construction method proposed by Lewko and Waters in [5]. This conversion restricts decryption to users who possess the corresponding attributes required to access the data.

As shown in Fig. 2, during the attribute matching phase, if the visitor's attributes satisfy a certain hierarchy level, they can decrypt the data file for that level. Only when the global access structure is fully satisfied can all data be accessed.

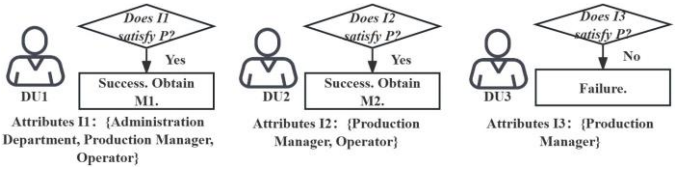

Figure 2. Hierarchical Access Control Example

III. Overview of the Proposed Scheme

The following sections introduce the system model and detailed procedures of the proposed scheme.

A. System Model

The proposed scheme involves six main entities: Data Owner (DO), Data User (DU), Cloud Service Provider (CSP), Central Authority (CA), Attribute Authority (AA), and Edge Proxy Node (EA). The system framework of the scheme is shown in Fig.3, with specific role definitions as follows.

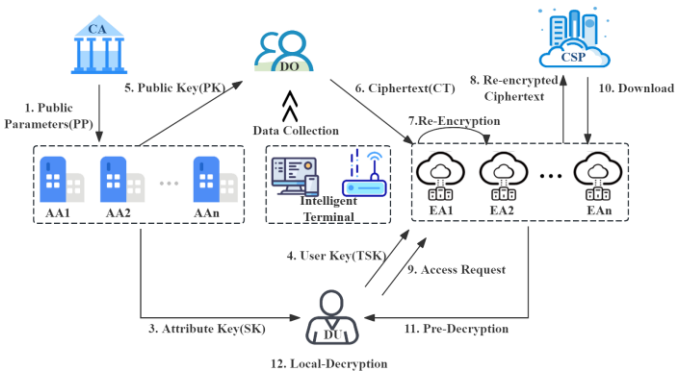

Figure 3. The system framework of the scheme

(1) Cloud Service Provider (CSP): The CSP is a semi-trusted entity responsible for storing the ciphertext uploaded by users. While the CSP may be curious about the data stored in the cloud, it is reliable and will strictly fulfill its designated services, refraining from maliciously deleting data or refusing user requests.

(2) Central Authority (CA): The CA is a fully trusted entity responsible for the initial setup of the system and the generation of public parameters (PP).

(3) Data Owner (DO): The DO is responsible for defining access policies and encrypting data based on these policies in collaboration with the CA before outsourcing it to the CSP.

(4) Data User (DU): The DU can obtain pre-decryption results from the edge nodes and then decrypt the data to access the plaintext.

(5) Edge Proxy Node (EA): The EA is an entity with storage and computation capabilities, capable of handling DU requests within its domain in real time. It can also retrieve ciphertext from neighboring domain proxy nodes or the cloud server as needed. Additionally, EA_s are responsible for re-encrypting and pre-decrypting ciphertexts.

(6) Attribute Authority (AA): The AA is also fully trusted. Each AA manages a unique set of attributes within its domain, ensuring that attributes managed by different AA_s do not overlap. The AA generates a public-private key pair for each attribute in its domain. The public key is sent to the DO for encryption, while the private key is used to generate attribute private keys for DU_s. The AA registers each user and maintains an attribute list (AL) bound to the user global identifier (UID) to verify the user's authenticity.

B. Algorithm Implementation

1) $Setup(\lambda) \rightarrow PP$

The algorithm is executed by the CA, with the security parameter λ as input, and outputs the system public parameters $PP = \{GF(q), E, G, U, H\}$, where $GF(q)$ represents a finite field of prime order q. E denotes an elliptic curve chosen over the finite field. G is a generator point on the elliptic curve E with order r. $U = \{attr_1, \dots\dots, attr_m\}$ represents the global set of system attributes managed by multiple attribute authorities, and

$H: \{0, 1\}^* \rightarrow Z_r^*$ is a hash function selected by the system to map the user global identifier UID to an element in Z_r.

2) $AASetup(PP) \rightarrow (PK, MSK, AL)$

The algorithm is executed by each Attribute Authority in the system's attribute domains, taking PP as input to run the $AASetup$ algorithm. This outputs the system's master secret key MSK, public key PK, and a record containing each user's UID along with a corresponding attribute list AL. The system has multiple AA_s, with each AA managing a unique, non-overlapping set of attributes. The algorithm is executed in two steps.

Step 1: For each attribute i in the system, randomly select integers $\alpha_i, k_i \in Z_r$ as part of the master secret key MSK, and compute $\{\alpha_i \cdot G, k_i \cdot G\}$ as part of the public key (PK). Thus, $MSK = \{\alpha_i, k_i, \forall i\}$, $PK = \{\alpha_i \cdot G, k_i \cdot G | \forall i\}$.

Step 2: For each device in the system, the AA negotiates a session key pair with the device during its registration with the system and maintains an attribute list AL associated with the user's UID.

3) $KeyGen(PP, I_{UID}, UID, MSK) \rightarrow SK_{i,UID}$

The algorithm is executed by AA_i with the system public parameters PP, user attribute set I_{UID}, user global identifier UID, and the master secret key MSK as input. It outputs the attribute private key corresponding to the UID, and records it in the corresponding AL.

For each attribute i bound to the UID, AA_i generates an attribute key $SK_{i,UID} = \alpha_i + H(UID) \cdot k_i$, where $SK = \{SK_{i,UID}, i \in I_{UID}\}$ is then sent to the DU and recorded in the AL.

4) $TransKey(SK_{i,UID}) \rightarrow TSK_{i,UID}$

This algorithm is executed by DU. Taking the attribute key $SK_{i,UID}$ as input, it outputs the user's private key $TSK_{i,UID}$.

Step 1: The DU selects a random number d as the user transformation key USK and computes $TSK_{i,UID} = SK_{i,UID} + USK = \alpha_i + H(UID) \cdot k_i + d$ as the final user private key.

Step 2: The DU sends $TSK = \{TSK_{i,UID}, i \in S_{j,UID}\}$ to the EA within the domain for safekeeping. This eliminates the need for the user to store the keys locally, thereby effectively conserving storage resources on the terminal device.

5) $File\ Encrypt(PP, PK, M, (A, \rho)) \rightarrow CT$

This algorithm is executed by DO. Taking as input the public parameters PP, the system public key PK, and a set of data files $M = \{M_1, \ldots \ldots, M_q\}$, it uses a hierarchical global access structure (A, ρ) to perform staged encryption on the data, resulting in the output ciphertext CT. Here, A is an $l \times n$ access matrix, and $\rho(i)$ represents the attribute corresponding to the ith row of the access matrix A.

Step 1: For each data file M_j at the jth level, the DO selects a random number $ck_i \in Z_r, i \in [1, q]$ as a symmetric key to perform the first-stage encryption on the data file. The encrypted file is computed as $C_{M_i} = Enc(M_i, ck_i)$. The DO then calculates $Mac_{M_i} = H(C_{M_i}) \cdot G$ to enable file integrity verification.

Step 2: Map the symmetric key ck onto a point on the elliptic curve. Select a random blinding factor $s_j \in Z_r, j \in [1, n]$ and the corresponding symmetric key ck_i for the jth level file will be encrypted with the blinding factor for the jth layer:

$$C_i = ck_i + s_j \cdot G$$

Step 3: Randomly select three column vectors: $\vec{v} = (s_1, s_2, \ldots, s_n) \in Z_r$, $\overrightarrow{u_j} = (u_1, \ldots, 0, \ldots, u_n) \in Z_r$ (where the jth element is 0), and $\vec{u} = (s_1 + s_2, \ldots, s_j + s_{j+1}, \ldots, s_n) \in Z_r$. Then, calculate $\lambda_x = \mathrm{A}_x \cdot \vec{v}$, $\omega_x = \mathrm{A}_x \cdot \overrightarrow{u_j}$, $r_x = \mathrm{A}_x \cdot \vec{u}$, $x \in [1, l]$.

Subsequently, The DO calculates the ciphertext components $C_{1,x}, C_{2,x}$ and $C_{3,x}$.

$$C_{1,x} = \lambda_x \cdot G + r_x \cdot \alpha_{\rho(x)} \cdot G$$
$$C_{2,x} = r_x \cdot G$$
$$C_{3,x} = \omega_x \cdot G + r_x \cdot k_{\rho(x)} \cdot G$$

The DO uploads the ciphertext $CT = \{C_{M_i}, Mac_{M_i}, (\mathrm{A}, \rho), C_i, C_{1,x}, C_{2,x}, C_{3,x}\}$ to the edge proxy node within the domain.

6) $Re - Encryption(CT) \rightarrow CT_{EA}$

This algorithm is executed by the EA. When the DO delivers the ciphertext CT to the EA within the domain, the EA, to enhance security during storage on the CSP, randomly selects an $EASK \in Z_r$ and computes $CT_{EA} = Enc(CT, EASK)$. The EA then uploads CT_{EA} to the CSP.

7) $File\ Decrtption$

Given the limited computational capacity of intelligent sensing terminals, the decryption process is divided into two phases: pre-decryption at the edge proxy and local decryption by the user DU. When DU submits an access request to the local EA, the EA retrieves the ciphertext CT_{EA} from the CSP. The edge proxy node then utilizes the retained TSK and $EASK$ to perform a preliminary decryption on CT_{EA} by executing the pre-decryption algorithm, and subsequently sends the partially decrypted result CT' to DU. Finally, DU obtains the correct information with a simple scalar multiplication.

Step 1: Identity Authentication

The DU must first undergo mutual authentication with the AA using the session key established during system registration. This step allows the AA to verify the $DU's$ identity by UID. If authentication fails, the decryption request is immediately denied, thereby preventing unnecessary computation.

Step2: $Pre - Decryption(PP, PK, TSK, ESAK, CT_{EA}) \rightarrow CT' or \perp$

(1) If identity authentication is successful, the EA first retrieves CT by decrypting CT_{EA} using the retained $EASK$, as $CT = Dec(CT_{EA}, EASK)$.

(2) The edge proxy EA then takes the $DU's$ attribute set I as input to generate the set $X = \{x | \rho(x) \in I\}$. If the DU satisfies part or all of the access policy, then for the jth level that is satisfied, there exists a set $\{c_{x,j} \in Z_r, x \in X, j \in [1, n-1]\}$ such that $\sum_{x \in X} c_{x,j} \cdot \mathrm{A}_x = \varepsilon_j$, where ε_j is a row vector of length n with the jth element as 1 and all other elements as 0. If the

access policy is not satisfied, the output is ⊥. For the jth level that is satisfied, the following is obtained:

$$\sum_{x\in X} c_{x,j} \cdot \lambda_x = s_j$$

$$\sum_{x\in X} c_{x,j} \cdot \omega_x = 0$$

$$\sum_{x\in X} c_{x,j} \cdot r_x = s_j + s_{j+1}$$

Then, calculate and decrypt the ciphertext components T_1 and T_2.

$$\begin{aligned} D_x &= C_{1,x} - TSK_{i,UID} \cdot C_{2,x} + H(UID)C_{3,x} \\ &= \lambda_x \cdot G - H(UID) \cdot \omega_x \cdot G - r_x \cdot d \cdot G \end{aligned}$$

$$T_1 = \sum_{x\in X} c_{x,j} \cdot D_x = s_j \cdot G - d \cdot (s_j + s_{j+1}) \cdot G$$

$$T_2 = \sum_{x\in X} c_{x,j} \cdot C_{2,x} = (s_j + s_{j+1}) \cdot G$$

Finally, the partially decrypted ciphertext $CT' = \{C_{M_i}, Mac_{M_i}, T_1, T_2, C_i\}$ is obtained, and the EA sends $CT'$to the local user DU.

Step3: $Local - Decryption(PP, USK, CT') \rightarrow M$

For the symmetric key ck_i' of the jth level data file, the intelligent sensing terminal user only needs to perform a single scalar multiplication operation using the user transformation key USK to retrieve it. This approach effectively conserves computational resources on the local device.

$$ck_i' = C_i - T_1 - USK \cdot T_2$$

After decrypting to obtain the jth level, the blinding factors for lower levels are embedded into the higher levels during encryption. As a result, the blinding factor for the $(j+1)th$ level can be derived using Equation (2). Consequently, for files at the $(j+1)th$ level, where $j \in [1, n-1]$, the lower-level files can be directly accessed.

$$ck_{i+1}' = C_{i+1} - \left(T_2 - s_j \cdot G\right)$$

Step 4: After obtaining ck_j', calculate $Mac_{M_j}' = H\left(C_{M_j}\right) \cdot G = H(Enc(M_j, ck_j')) \cdot G$. If $Mac_{M_j}' = Mac_{M_j}$, this indicates that the data has not been maliciously tampered with. The DU can then use the symmetric key ck_j' to perform symmetric decryption and retrieve the file M_j, where $M_j = Dec(C_{M_j}, ck_j')$.

IV. Security Discussion and Analysin

A. Security Analysis

This section demonstrates that the proposed multi-authority CP-ABE hierarchical access control scheme for the Industrial Internet of Things is secure under the Decisional Diffie-Hellman assumption.

Theorem 1. The proposed scheme is indistinguishable under the Decisional Diffie-Hellman (DDH) assumption in the standard model when subjected to an adaptive chosen-ciphertext attack.

Proof: Assume that there exists an adversary $\mathcal{A}$ with a non-negligible advantage ε in breaking the proposed scheme. Then, there exists an algorithm $\mathcal{B}$ that can solve the DDH problem with advantage ε. Algorithm $\mathcal{B}$ is provided with the DDH challenge tuple (G, aG, bG, R), where its task is to distinguish whether $Z = abG$ or $Z = R$, with R being a random element in group G_E. Assume that the adversary $\mathcal{A}$ can query keys and win the game with a non-negligible advantage $\varepsilon > 0$. However, a restriction is imposed such that the queried private keys still do not satisfy the access structure. Under this constraint, the security game for the multi-authority scheme can be considered equivalent to that of the single-authority scheme. Algorithm $\mathcal{B}$ then plays the role of challenger $\mathcal{C}$, simulating the identities and playing a game with adversary $\mathcal{A}$ as follows.

1) Initialization

The adversary $\mathcal{A}$ selects a challenge access structure (A, ρ) and sends it to the simulator $\mathcal{B}$ as the DDH challenge.

2) Setup

The simulator $\mathcal{B}$, using the public parameters PP initialized by the challenger $\mathcal{C}$, selects two random values $\alpha_i, k_i \in Z_r$ for each attribute i defined in the system. It then generates the public key $PK = \{\, \alpha_i \cdot G, k_i \cdot G | \forall i \,\}$ and makes it publicly available.

3) Phase 1

The adversary $\mathcal{A}$ can adaptively submit queries (i, UID) to the simulator $\mathcal{B}$ to obtain the attribute decryption key corresponding to attribute i. The only restriction is that $\mathcal{A}$ cannot request a set of attribute decryption keys that would enable successful decryption. In response to $\mathcal{A}$'s request, $\mathcal{B}$ records the attribute in the attribute list associated with $\mathcal{A}$'s identity UID and selects a random $d \in Z_r$, calculating $SK_{i,UID} = \alpha_i a + H(UID) \cdot k_i + d$ as the attribute decryption key for attribute i requested by $\mathcal{A}$.

4) Challenge

The adversary $\mathcal{A}$ selects two messages $m_0$and $m_1$of equal length and submits them to the simulator $\mathcal{B}$ The simulator $\mathcal{B}$ performs a coin toss to randomly select $\beta = \{0,1\}$ and executes Algorithm 5) from Section III to encrypt the plaintext message m_β(To avoid ambiguity, only a single piece of data is encrypted in this case.).

Finally, the simulator $\mathcal{B}$ returns the challenge ciphertext $CT = \{C_{M_j}, Mac_{M_j}, (\mathrm{A}, \rho), C_j, C_{1,x}, C_{2,x}, C_{3,x}\}$ to the adversary $\mathcal{A}$.

5) Phase 2

The adversary $\mathcal{A}$ can continue to submit queries (i, UID) to the simulator $\mathcal{B}$ to obtain the attribute decryption key corresponding to attribute i. The only restriction remains the same as in Phase 1.

6) Guess

The adversary $\mathcal{A}$ outputs its guess β' for β. If $\beta' = \beta$, then the simulator $\mathcal{B}$ outputs 0, indicating that it guesses $Z = abG$; otherwise, $\mathcal{B}$ outputs 1, indicating that it guesses $Z = R$. If $Z = abG$, then this represents a real ciphertext, and the adversary $\mathcal{A}$ has an advantage as defined in the assumption, denoted by ε. Therefore, in this case, the probability that $\mathcal{A}$ correctly guesses β is $\Pr[\mathcal{B}(G, aG, bG, Z = abG) = 0] = \frac{1}{2} + \varepsilon$.

If $Z = R$, since R is randomly selected, the adversary $\mathcal{A}$ gains no useful information about β. In this case, the probability that $\mathcal{A}$ correctly guesses β is $\Pr[\mathcal{B}(G, aG, bG, Z = R) = 0] = \frac{1}{2}$.

Thus, the advantage of the simulator $\mathcal{B}$ in solving this security problem is.

$$\mathcal{B} = \frac{1}{2}(\Pr[\mathcal{B}(G, aG, bG, Z = abG) = 0] + \Pr[\mathcal{B}(G, aG, bG, Z = R) = 0]) - \frac{1}{2} = \frac{1}{2}\left(\frac{1}{2} + \varepsilon + \frac{1}{2}\right) - \frac{1}{2} = \frac{\varepsilon}{2}$$

Since the adversary $\mathcal{A}$ has a non-negligible advantage as assumed, the simulator $\mathcal{B}$ also has a non-negligible advantage. Thus, it can be concluded that the proposed scheme is secure under the DDH assumption.

B. Security Discussion

1) Data Confidentiality

The proposed scheme, based on the ECDLP, ensures that users who do not possess the required attributes cannot derive any information about their private keys $\{\alpha_i, k_i\}$ from the public key $\{\alpha_i \cdot G, k_i \cdot G\}$. Additionally, the data M is encrypted using the symmetric key ck, which is implicitly contained in the ciphertext C_{M_j}. Assuming the symmetric key ck can be mapped to cG, where $c \in Z_r$, and given that s_j is randomly selected by the data owner, $C_{M_j} = (\mathrm{c} + s_j) \cdot G$ becomes a random point on the elliptic curve. Based on the ECDLP, an adversary cannot gain any valuable information about ck or M.

Furthermore, s_j is secret-shared into l parts through λ_x, and only users with sufficient keys TSK can recover the corresponding hierarchical level. Thus, ciphertexts at a specific hierarchical level can only be decrypted if the access structure (A, ρ) satisfies the data user's attributes. Unauthorized users who lack the attributes corresponding to rows A_x in the matrix A cannot compute the blinding factors for the corresponding level in the vector $\vec{v}$. Therefore, the proposed scheme guarantees the security of the data.

Finally, each AA_s generates a portion of the keys within its attribute domain, and the EA selects the re-encryption key before uploading the ciphertext to the CSP. Consequently, the AA, CA, and CSP cannot access the original message. Moreover, the EA cannot fully decrypt the ciphertext CT without possessing the data user's transformation key USK.

2) Collusion Attack

Firstly, collusion between different data users must be addressed, meaning the scheme needs to prevent multiple users, who cannot independently decrypt, from collaborating and sharing attributes to obtain the complete set of attributes required for decryption. In the proposed scheme, both the user's private key and transformation key generated during the key generation phase are bound to the specific user's attributes $\rho(i)$ and unique identifier UID. This binding ensures that the system can verify whether all the keys in the attribute key set originate from the same user by checking the identifiers embedded in the keys. If multiple different UID values are detected during the attribute matching phase, it will be impossible to satisfy $\sum_{x \in X} c_{x,j} \cdot \lambda_x = s_j$, and consequently, $s_j \cdot G$ cannot be recovered. This indicates collusion among users, and decryption will be denied.

The second type of collusion attack involves collaboration between the EA and a data user. If the edge node sends the partially decrypted result CT' to an unauthorized user UID', the user UID' must still use the USK to complete the final decryption. However, the transformation key USK is generated using a random number d, which is unique to each user. Since the random number d differs for each user, their private keys are also distinct. An incorrect random number d will prevent UID' from successfully decrypting the partially decrypted result CT', making it impossible to retrieve the plaintext M.

From the analysis of these two types of collusion attacks, it can be concluded that the proposed scheme is resistant to collusion attacks.

V. Performance Analysis

In this section, we conduct a performance evaluation of the previously proposed schemes by Wang [9], He [14], Ding [16], S. Das [20], and our proposed scheme from both theoretical analysis and experimental validation perspectives. The evaluation focuses on comparing the security features computational cost of each scheme to assess their overall efficiency.

A. Theoretical Analysis

The security features of the proposed scheme, compared with other existing schemes, are summarized in Table I. Both schemes [9] and [14] support hierarchical data sharing for files. However, the tree structure used in scheme [9], which incorporates secret factors, lacks flexibility in semantic expressiveness. While scheme [14] supports the LSSS structure, neither of these schemes is pairing-free, and both rely on a single authority center for key generation, leading to challenges in terms of security and system efficiency.

The schemes proposed in [16] and [20] are ECC-based, pairing-free, and adopt the LSSS access policy, offering rich and flexible expressiveness. However, scheme [13] still employs a single authority for attribute distribution and management and does not support outsourced decryption. This limitation makes it more vulnerable to attacks in distributed IIoT environments, potentially leading to system downtime. Although scheme [20] incorporates multiple attribute authorities and outsourced decryption modules, both schemes [16] and [20] lack support for hierarchical file data with subordinate relationships. Consequently, the processing of hierarchical files in industrial IoT environments requires repetitive and unnecessary multi-round computations.

In contrast, the proposed scheme utilizes the LSSS structure, embedding blinding factors for multiple hierarchical files to enable collaborative encryption, while eliminating the need for pairing operations. Additionally, it employs multi-authority key management and outsources the decryption phase through edge computing technology. Evidently, the proposed scheme is more suitable for resource-constrained industrial IoT environments.

TABLE I. COMPARISON OF THE SCHEMES.

Scheme	[9]	[14]	[16]	[20]	Our
Pairing Free	No	No	Yes	Yes	Yes
Access Structure	Tree	LSSS	LSSS	LSSS	LSSS
Multi-Authority	No	No	No	Yes	Yes
Decryption Outsourcing	No	No	No	Yes	Yes
Hierarchical Access	Yes	Yes	No	No	Yes

TABLE II. SYMBOL DESCRIPTIONS

Symbols	Meanings of Symbols
N_c	Number of Attributes in the Access Structure
N_a	Minimum Number of Attributes Required to Satisfy Access Policy n
q	Number of Files
n	Number of Hierarchical Levels in the Access Structure
T_G	Exponentiation Operation in Group G
T_{G_T}	Exponentiation Operation in Group G_T
T_{pair}	Bilinear pairing operation
T_{ECC}	Scalar Multiplication Based on ECC

Assume there is a set of files consisting of q files that need to be encrypted and decrypted, where their corresponding access structures exhibit a subordinate relationship, i.e., $P_q \subset P_{q-1} \subset \cdots \subset P_1$. Before conducting the computational overhead analysis, we first define the symbols used in the theoretical analysis of the scheme in Table II.

The computational cost comparison between our scheme and the schemes [14], [16], and [20] primarily involves three aspects: encryption by the data owner, decryption by the edge proxy node, and local decryption by the user. As shown in Table III, both our scheme and [14] incorporate the concept of file hierarchy, utilizing a global LSSS structure that carries blinding factors for multiple hierarchical files. This significantly reduces the computational cost compared to schemes [16] and [20], which require the setup of q access policies and q rounds of encryption and decryption. Meanwhile, our scheme, as compared to [14], uses scalar multiplication based on ECC and employs edge computing for outsourced decryption. As a result, the user only needs to perform a single scalar multiplication locally to complete the decryption process.

TABLE III. COMPARISON OF COMPUTATION OVERHEAD OF THE SCHEMES.

Scheme	Encryption	Pre-Decryption	Local Decryption
[14]	$3(N_{c_1}+n)T_G + nT_{G_T}$	/	$N_{a_1}T_{G_T} + (2N_{a_1}+1)T_{pair}$
[16]	$\{3(N_{c_1}+\cdots+N_{c_q})+q\}T_{ECC}$	/	$\{2(N_{a_1}+\cdots+N_{a_q})\}T_{ECC}$
[20]	$\{4(N_{c_1}+\cdots+N_{c_q})+2q\}T_{ECC}$	$\{3(N_{a_1}+\cdots+N_{a_q})+1\}T_{ECC}$	$(q)T_{ECC}$
Our	$\{4N_{c_1}+2q\}T_{ECC}$	$\{3N_{a_1}+1\}T_{ECC}$	$(1)T_{ECC}$

B. *Experimental Analysis*

To more intuitively demonstrate the advantages of the proposed scheme, we conducted simulation experiments on Ubuntu 20.04 x64. The server used for the experiments was equipped with a 2.1 GHz 12th Gen Intel(R) Core(TM) i7-1260P processor and 16GB RAM. The experimental code was written using the charm-crypto framework and Python 3.8, and the experiments were conducted on a super singular curve $y^2 = x^3 + x$ over a 512-bit finite field, utilizing a 160-bit elliptic curve group. The time comparisons for performing various operations in this environment are shown in Table III. The experimental results represent the averages of 30 rounds of testing.

In the following computation time comparison, the following assumptions are made for the upcoming computation time comparison: Since the access structure in our proposed scheme only contains a monotonic access structure with "AND" and "OR" gates, in the definition of the hierarchical access tree, besides the leaf nodes that represent attributes, each child node is either a transmission node (if a node has at least one restricted gate in its child nodes, then the node is considered a transmission node) or a level node. Each transmission node has two child nodes. Under the conditions of the access structure, we set the number of files equal to the number of hierarchical levels in the access structure, which represents the worst-case scenario for the efficiency of the proposed scheme.

In the actual simulation, we assume that the number of attributes is $|N| = 20$, and the number of files considered are 2, 4, 6, 8, 10, 12, and 14. The specific computational overhead for encryption, pre-decryption, and local decryption is shown in Fig.4, 5, and 6, respectively.

Fig.4 shows that as the number of files increases, the encryption time for all four schemes increases. However, the schemes supporting hierarchical access structures, namely He [14] and our scheme, exhibit a much smaller growth rate compared to Ding [16] and S. Das [20]. Moreover, since our scheme does not involve bilinear pairing operations, it demonstrates a clear advantage in terms of time efficiency. As the number of hierarchical levels increases, the advantage of our scheme becomes even more pronounced.

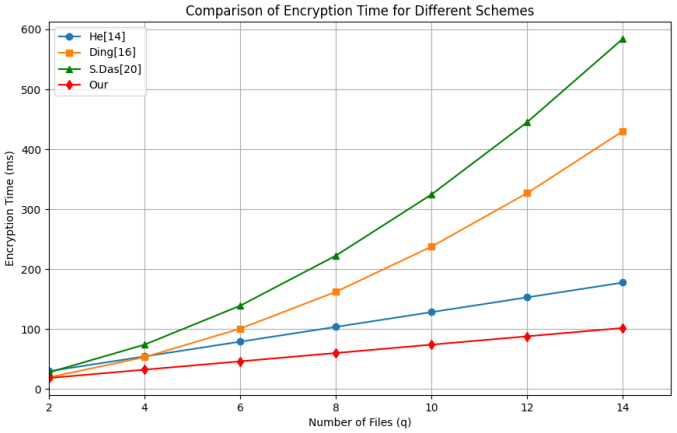

Figure 4. Comparison of Encryption Time for Different Schemes

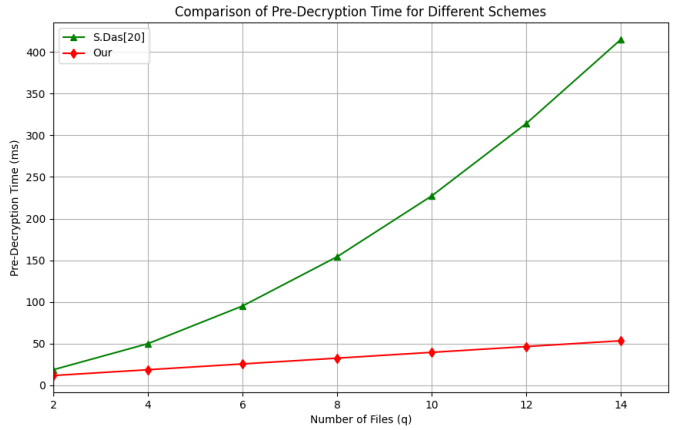

Figure 5. Comparison of Pre-Decryption Time for Different Schemes

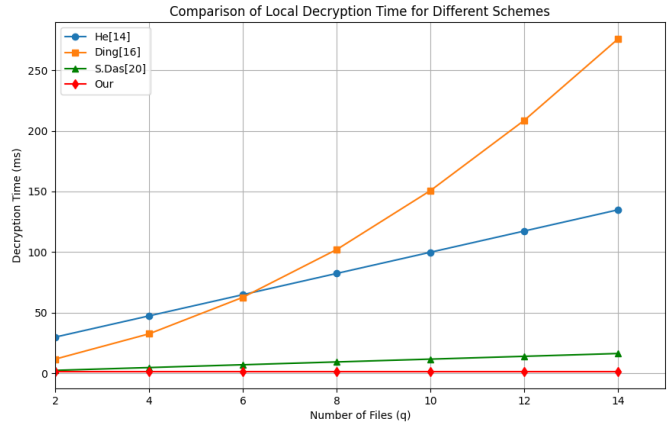

Figure 6. Comparison of Local Decryption Time for Different Schemes

From Fig.5, it can be observed that in the two schemes supporting outsourced decryption, the outsourced decryption time in our scheme increases linearly with the number of files, whereas the growth rate in the Das S [20] scheme becomes more pronounced as the number of files increases. This is due to our scheme using a global LSSS structure to support hierarchical sharing, which gives it a computational advantage.

As shown in Fig.6, regarding local decryption, since both our scheme and the S. Das [20] scheme offload part of the decryption task to edge nodes, the local decryption time does not significantly increase with the complexity of the access policy. Additionally, compared to the S. Das [20] scheme, our scheme incurs lower decryption costs, making it more suitable for resource-constrained terminal devices.

VI. Conclusions

In this article, a multi-authority CP-ABE hierarchical access control scheme based on ECC is proposed. The scheme utilizes edge computing to decouple the decryption phase, ensuring the inherent security of industrial systems while fundamentally improving overall operational efficiency. By using a global LSSS to carry multiple data files with hierarchical access relationships, when the attributes of a data requester match a portion of the access control structure, a single decryption operation can provide the corresponding file data for that portion. This approach presents a lightweight and efficient collaborative data security solution for the Industrial Internet of Things. Furthermore, to address the issues of single authority and key management, multiple attribute authorities are employed to manage user attributes, reducing the workload of the central authority. Finally, both theoretical and experimental analyses demonstrate that our scheme is effective and suitable for IIoT applications.

However, our scheme still has certain limitations, such as not accounting for the potential privacy leakage caused by the access policy itself revealing user-related information. In the future, we plan to explore methods for hiding access policies to achieve better privacy protection.

Acknowledgment

This work is supported by National Key R&D Program of China (No.2021YFB3101900).

References

[1] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, pp. 457-473, 2005.

[2] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption”, in Proceedings of the 2007 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 20–23 May 2007, pp. 321-334, 2007.

[3] M. Chase, “Multi-authority Attribute Based Encryption”, in Theory of Cryptography, S. P. Vadhan, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 515–534.

[4] A. Lewko and B. Waters, “Decentralizing Attribute-Based Encryption”, in Advances in Cryptology – EUROCRYPT 2011, K. G. Paterson, Ed., Berlin, Heidelberg: Springer, 2011, pp. 568–588.

[5] Y. Yang, X. Chen, H. Chen, and X. Du, “Improving Privacy and Security in Decentralizing Multi-Authority Attribute-Based Encryption in Cloud Computing”, IEEE Access, vol. 6, pp. 18009–18021, 2018.

[6] M. Gupta, F. M. Awaysheh, J. Benson, M. Alazab, F. Patwa, and R. Sandhu, “An Attribute-Based Access Control for Cloud Enabled Industrial Smart Vehicles”, IEEE Transactions on Industrial Informatics, vol. 17, no. 6, pp. 4288–4297, Jun. 2021.

[7] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia, “PHOABE: Securely outsourcing multi-authority attribute based encryption with

policy hidden for cloud assisted IoT", Computer Networks, vol. 133, pp. 141–156, Mar. 2018.

[8] H. Zhong, Y. Zhou, Q. Zhang, Y. Xu, and J. Cui, "An efficient and outsourcing-supported attribute-based access control scheme for edge-enabled smart healthcare", Future Generation Computer Systems, vol. 115, pp. 486–496, Feb. 2021.

[9] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "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, Jun. 2016.

[10] G. K. Sandhia, S. V. Kasmir Raja, and K. R. Jansi, "Multi-Authority-Based File Hierarchy Hidden CP-ABE Scheme for Cloud Security", SOCA, vol. 12, no. 3, pp. 295–303, Dec. 2018.

[11] J. Fu and N. Wang, "A Practical Attribute-Based Document Collection Hierarchical Encryption Scheme in Cloud Computing", IEEE Access, vol. 7, pp. 36218-36232, 2019.

[12] P. S. Challagidad and M. N. Birje, "Efficient Multi-authority Access Control using Attribute-based Encryption in Cloud Storage", Procedia Computer Science, vol. 167, pp. 840–849, Jan. 2020.

[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, Apr. 2021.

[14] H. He, L. Zheng, P. Li, L. Deng, L. Huang, and X. Chen, "An efficient attribute-based hierarchical data access control scheme in cloud computing", Hum. Cent. Comput. Inf. Sci., vol. 10, no. 1, p. 49, Dec. 2020.

[15] V. Odelu and A. K. Das, "Design of a new CP-ABE with constant-size secret keys for lightweight devices using elliptic curve cryptography", Security and Communication Networks, vol. 9, no. 17, pp. 4048–4059, 2016.

[16] S. Ding, C. Li, and H. Li, "A Novel Efficient Pairing-Free CP-ABE Based on Elliptic Curve Cryptography for IoT", IEEE Access, vol. 6, pp. 27336–27345, 2018.

[17] A. K. Junejo and N. Komninos, "A Lightweight Attribute-Based Security Scheme for Fog-Enabled Cyber Physical Systems", Wireless Communications and Mobile Computing, vol. 2020, no. 1, p. 2145829, 2020.

[18] Y. Tian, T. Shao, and Z. Li, 'An Efficient Scheme of Cloud Data Assured Deletion', Mobile Netw Appl, vol. 26, no. 4, pp. 1597–1608, Aug. 2021.

[19] R. Cheng, K. Wu, Y. Su, W. Li, W. Cui, and J. Tong, "An Efficient ECC-Based CP-ABE Scheme for Power IoT", Processes, vol. 9, no. 7, Art. no. 7, Jul. 2021.

[20] S. Das and S. Namasudra, "MACPABE: Multi-Authority-based CP-ABE with efficient attribute revocation for IoT-enabled healthcare infrastructure", International Journal of Network Management, vol. 33, no. 3, p. e2200, 2023.

[21] Y. Liu, S. Xu, and Z. Yue, "A Lightweight CP-ABE Scheme with Direct Attribute Revocation for Vehicular Ad Hoc Network", Entropy, vol. 25, no. 7, Art. no. 7, Jul. 2023.