Computer Science and Information Systems 23(2):885–916 https://doi.org/10.2298/CSIS260208016Z

FPCA: A Fully Constant-Length and Policy Updating Cross Data Domain Access Control for Cloud-Edge Collaborative Environment

Shiwen Zhang[1,2], Siwei Wen[1,2], and Wei Liang[1,2]

[1] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China
[2] Sanya Research Institute, Hunan University of Science and Technology, Sanya 572024, China
shiwenzhang@hnu.edu.cn (corresponding author)
siweiwen@mail.hnust.edu.cn
wliang@hnust.edu.cn

Abstract. The secure data sharing across data domains in cloud-edge collaborative environment faces challenges of low data access control efficiency and a lack of dynamism. In this work, we proposed FPCA, a fully constant-length and policy updating cross data domain access control for cloud-edge collaborative environment. FPCA address the above challenges by proposing three algorithms: the Multi Data Domains Key Generation (MDKG) algorithm maintains constant secret key length and enables cross data domain access without attribute conversion, the Constant-Length Ciphertext Encryption (CLCE) algorithm maintains constant ciphertext length, reducing decryption overhead, and the Access Policy Update (APU) algorithm updates the access policy with constant-length update message and low computational complexity. Security analysis demonstrates FPCA resist key forgery, collusion, and chosen-key attacks while ensuring backward security. Simulation experiment shows that FPCA outperforms existing schemes. For FPCA, the storage costs are only 50% to 80% of other constant-length ABE schemes, the communication costs remains at 0.38–1.27 KB between entities, the time costs of access policy update and cross data domain access remain around 8.4 ms and 30.4 ms, respectively, and are lower than other similar schemes. These results confirm the efficiency of FPCA in achieving secure and dynamic cross data domain access in cloud-edge collaborative environment.

Keywords: Cross Data Domain Access Control, Constant-length Secret Key, Constant-length Ciphertext, Access Policy Update, Cloud-edge Collaborative Environment.

1. Introduction

Cloud-edge collaboration technology has promoted the development of the Internet of Things (IoT), improving the scale and efficiency of data sharing across different data domains. By 2030, the total number of IoT devices worldwide is expected to reach 50 billion [13]. Cloud servers provide powerful data storage services, edge servers can preload data from cloud servers, and users interact with nearby edge servers to share or retrieve data of interest from any data domain [19]. Cloud-edge collaboration technology provides low-cost and low-latency data sharing services for IoT users across multiple data domains [26]. Unfortunately, data uploaded directly to cloud or edge servers may pose

security risks of privacy leakage [3, 29]. For this, access control technology is essential to ensure secure data sharing between multiple data domains [14]. Based on the issue of privacy leakage, many works on data access control have been proposed [15–18]. However, these schemes have a huge key management burden and do not provide fine-grained access control, which is unsuitable for a cloud-edge collaborative environment with many IoT users.

To realize fine-grained data access control, Attribute-Based Encryption (ABE) [21,22] has been proposed. ABE provides fine-grained access control by setting attributes and formulating access policies [5, 22]. In cloud-edge collaborative environments, the data that users need to access usually comes from different data domains (Multiple data domains belong to the same trust domain, with different attribute sets between them) [11,19]. However, traditional ABE does not support data access across multiple data domains, and the length of the ciphertext and secret key increases as the number of attributes increases [21], generating massive consumption of computing resources for both servers and users, and reducing the efficiency of data access control [3]. In addition, in the cloud-edge collaborative environment, the Internet of Things must support dynamic user management to adapt to flexible access control permission change requirements, while traditional ABE cannot dynamically adjust user access control permissions. Thus, developing an efficient and dynamic cross data domain access control mechanism for the cloud-edge collaborative environment is essential.

To realize data access control across multiple data domains, several schemes use attribute conversion technology to achieve cross data domain access control [4, 20]. For example, [20] establishes conversion relationships between different attribute sets in different data domains that can convert its own secret key to a valid secret key in the target data domain. Nevertheless, there are different conversion methods between different attributes, which require the storage of multiple conversion keys and bring a high cost of attribute conversion calculation to IoT users. In addition, some attributes may not have a conversion relationship between them, resulting in inflexible key conversion across data domains. [4] consolidates the attributes of multiple data domains into a secret key and [10] embeds the attribute information from multiple data domains in ciphertext to eliminate the attribute conversion operation. However, multiple data domains contain a large number of attributes that significantly increase the length of ciphertexts and secret keys, which also burdens IoT users. [9] can keep the length of the secret key and the ciphertext constant. Unfortunately, this scheme cannot support cross data domain access control. Therefore, maintaining constant ciphertext and secret key length while reducing computation across multiple data domains in cross data domain access control to improve access control efficiency remains a challenge.

To achieve dynamic user access control, many schemes have been designed to update access policies [12, 25]. For example, [25] sets the version number for ciphertext and secret key, and uses a re-encryption key to update the access policy. This scheme requires updating both the ciphertext and the secret keys of all unreleased users simultaneously. [12] uses the Attribute Cuckoo Filter (ACF) to dynamically adjust access policies and only needs to update the ciphertext. However, this scheme needs to update the version number of each ciphertext component and conduct secret sharing of the ciphertext again, equivalent to recalculating the ciphertext. [6] and [30] divided the attributes in the access policies into different types, which can reduce the calculation of the old attribute

part in the ciphertext. However, in order to prevent adversaries from using old secret values to decrypt ciphertext, the secret value of the ciphertext needs to be reselected and the secret shared again, which still requires a lot of computation and makes the efficiency of dynamic access control remain low. Therefore, designing an efficient dynamic access control mechanism with low computational complexity to update the access policy remains a pressing challenge.

In this work, we propose an efficient and dynamic cross data domain access control scheme with policy update for a cloud-edge collaborative environment. To minimize the consumption of computing resources for cross data domain access control, we designed the Multi Data Domains Key Generation (MDKG) algorithm and the Constant-Length Ciphertext Encryption (CLCE) algorithm to achieve efficient cross data domain access control. In MDKG and CLCE, we encode and polymerize attributes across multiple data domains by attribute encoding summation and group element multiplication that can reduce and maintain the length of the secret key and the ciphertext, respectively, which can reduce the computational resource consumption for IoT users. To achieve dynamic user access control, we also designed an Access Policy Update (APU) algorithm that updates the access policy with constant-length update message and low computational complexity. In summary, the contributions of this work are as follows:

1. We propose FPCA, a fully constant-length and policy updating cross data domain access control for cloud-edge collaborative environment, which achieves efficient, fully constant-length cross data domain access control and can update the access policy with low computational complexity.
2. We propose the MDKG algorithm to improve the efficiency of cross data domain access control. The MDKG keeps the secret key length constant, allowing IoT users to efficiently access data from other data domains without any conversion operation.
3. We propose the CLCE algorithm to reduce the computational resource costs of IoT users and servers, keeping the ciphertext length constant and reducing the decryption calculations for IoT users.
4. We propose the APU algorithm to achieve dynamic access control, updating the access policy with constant-length update message and low computational complexity.

The remainder of this article is organized as follows. Section 2 discusses related work relevant to our proposed research, while the preliminary, system model, and threat model are introduced in Section 3. The multiple data domain model, the proposed FPCA, and the correctness analysis are presented in Section 4. We perform a security analysis in Section 5. We analyze and evaluate the performance of our FPCA in Section 6, and finally, concluding remarks and future directions are presented in Section 7.

2. Related Work

2.1. Cross Data Domain Access Control

To address the problem of data access across multiple data domains, Huai *et al.* [8] propose a data sharing scheme across data domains. However, this scheme has system security issues and may be threatened by various attacks during data sharing. For better security, many schemes [20, 23] proposed blockchain-based data access schemes across data

domains. Using the immutability of blockchain to ensure security. Sun *et al.* [20] establishes a conversion relationship between the attributes of each data domain and converts the attributes in the secret key of the IoT user to other attributes in the corresponding data domain through the conversion relationship. However, these schemes need to perform a large number of attribute conversion operations for cross data domain access, which imposes a heavy burden on IoT users and reduces the efficiency of access control. Fan *et al.* [6] consolidates the attributes of different data domains into a single secret key, allowing users to access data from different data domains with a single secret key. In addition, Li *et al.* [10] consolidates the attributes of different data domains into one ciphertext, allowing the ciphertext of a certain data domain to be decrypted by users of other different data domains without the need for additional attribute conversion operations by users. However, the length of the secret key and the ciphertext in these schemes will increase with the number of attributes and data domains, reducing the efficiency of access control. Yannis *et al.* [27] proposed an attribute compression scheme that reduces the number of attribute-related components in both the ciphertext and secret key, thus reducing their lengths. However, the lengths of the ciphertext and secret key still increase with increasing number of attributes. Thus, some constant-size ABE schemes [1,2,7] have been proposed. Allison *et al.* [1] and Wu *et al.* [2] can achieve a constant-length ciphertext, but the length of the secret key still increases with the number of attributes. Fan *et al.* [7] can achieve both the length of the ciphertext and the secret key constant, improving the efficiency of access control. However, these schemes cannot achieve cross data domain access control. Therefore, we design an efficient cross data domain access control scheme that removes the attribute conversion operation and keeps both the secret key length and the ciphertext length constant to improve the efficiency of data access control across multiple data domains.

2.2. Access Policy Update

To improve the efficiency of access policy update, many schemes [10, 28] use the re-encryption key as the update key to update the ciphertext. Xue *et al.* [24] proposed an access policy update scheme for hidden access policies, improving the security of the access policy update. However, the computational cost of the update calculation in these schemes increases with increasing number of attributes in the ciphertext. Fan *et al.* [6] divided the attributes in access policies into three types that can reduce the computation of parts of the ciphertext that do not require updates. Thus, reduces the computational complexity of the update of the access policy. Zuo *et al.* [30] proposed a blockchain-based access policy update scheme that also divided attributes of access policies into different types to improve the efficiency of policy update and use the blockchain to improve security. However, these schemes cannot maintain a constant length of the ciphertext, and the computational cost of updating the old ciphertext increases as the number of attributes in the ciphertext increases. Additionally, when updating the access policy for ciphertext, to prevent adversaries from decrypting the updated ciphertext with old decryption results, it is necessary to set a new secret value and perform the secret sharing calculations again, resulting in low update efficiency of the access policy and reduced flexibility of access control. Fan *et al.* [7] add a timestamp to user's attribute set and access policy. By updating the timestamps in the access policy, [7] can significantly reduce the overhead of

updating the access policy. However, it cannot modify attribute related content in the access policy except for timestamps. Additionally, it fails to update the secret value in the ciphertext, thus failing to meet the backward security. Therefore, we propose an efficient and secure policy update access control scheme that implements the update calculation with complexity $O(1)$ and keeps the length of the update message constant to improve the efficiency and flexibility of the data access control.

In Table 1, we compared the features between FPCA and existing related works.

Table 1. Feature Comparison on FPCA and Other Related Works

Scheme	DAS	CDD	AC	CCL	CSKL	APU	BS	CUM
[1]	Y	N	N	Y	N	N	N	N
[2]	Y	N	N	Y	N	N	N	N
[6]	Y	Y	N	N	N	Y	Y	N
[7]	Y	N	N	Y	Y	Y	N	Y
[8]	N	Y	N	N	N	N	N	N
[10]	Y	Y	N	N	N	Y	Y	N
[20]	Y	Y	N	N	N	N	N	N
[23]	Y	Y	N	N	N	N	N	N
[24]	Y	N	N	N	N	Y	Y	N
[27]	Y	N	Y	N	N	N	N	N
[28]	Y	N	N	N	N	Y	Y	N
[30]	Y	N	N	N	N	Y	Y	N
FPCA	Y	Y	Y	Y	Y	Y	Y	Y

Notes: DAS: Data Secrurity. CDD: Cross Data Domain Access Control. AC: Attribute Compression. CCL: Constant Ciphertext Length. CSKL: Constant Secret Key Length. APU: Access Policy Update. BS: Backward Security. CUM: Constant Update Message Length.

3. Problem Formulation

3.1. Preliminary

Selective game for ABE: The selective game described in the following involves an adversary A and a challenger CH.

1. ***Init***: A chooses an access policy $\mathfrak{R}_1$ and sends it to CH.
2. ***Setup***: CH runs the **System Initialization** algorithm to generate the public parameters PP and the master secret key MK, CH sends PP to A.
3. ***Key Query 1***: A submits the attribute set U_A to CH to request the secret key, U_A cannot satisfy $\mathfrak{R}_1$. CH runs the MDKG algorithm to generate the secret key SK_A based on U_A and sends SK_A to A.
4. ***Challenge***: A sends two messages of equal length $\{M_0, M_1\}$ to CH. CH randomly chooses $\beta \in \{0, 1\}$ and runs the CLCE algorithm to encrypt M_β based on $\mathfrak{R}_1$ and returns the challenge ciphertext CT_c to A.

5. ***Update Ciphertext***: A generates a new access policy $\mathfrak{R}_2$, $\mathfrak{R}_2$ cannot be satisfied by any U_A submitted by A. A sends $\{CT_c, \mathfrak{R}_2\}$ to CH. CH runs the APU algorithm and returns the updated ciphertext CT_{new} based on $\mathfrak{R}_2$ to A.
6. ***Key Query 2***: Repeat the process of **Key Query 1**, but none of the aforementioned secret keys can decrypt CT_c or CT_{new}.
7. ***Guess***: A outputs $\beta^{'} \in \{0, 1\}$ as a guess of β. If $\beta^{'} = \beta$, A wins the security game. The advantage of A to win the security game is defined as $\Omega = Pr[A\ win] - 1/2$.

3.2. System Model

The system model of our FPCA is illustrated in Fig. 1, which involves six participants: 1) **Center Authority**, 2) **Attribute Authority**, 3) **Edge Server**, 4) **Cloud Server**, 5) **Data Owner**, and 6) **Data User**. Their main functions are depicted below.

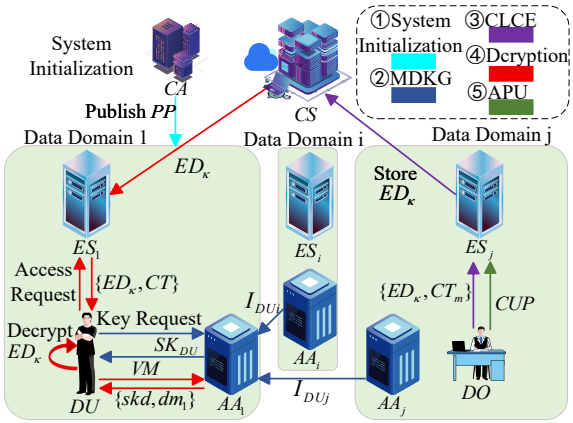


Fig. 1. The System Model of FPCA

1. **Center Authority** (CA)**:** CA is responsible for performing the System Initialization algorithm to generate the public parameters and the master secret key. CA publishes the public parameters for each entity. CA assigns the attribute set to each attribute authority. CA is trusted.
2. **Attribute Authority** (AA)**:** Each AA controls a disjoint attribute subset and is responsible for generating the secret key for the data user. AAs are also responsible for checking if the data user satisfies the access policy. AAs are trusted.
3. **Edge Server** (ES)**:** ES is responsible for the storage of the ciphertext. ES is semi-trusted.
4. **Cloud Server** (CS)**:** CS is only responsible for storing encrypted data.
5. **Data Owner** (DO)**:** DO is an IoT user. DO is responsible for developing and updating access policies. DO is also responsible for encrypting the data to be shared and generating the ciphertext according to the access policy.

6. **Data User (DU):** DU is an IoT user. DU gets the secret key from AA according to its attributes set. DU can obtain any ciphertext of interest from ES. Only DU that comply with the access policy can correctly decrypt the ciphertext using its secret key.

There are multiple data domains in the system, each data domain contains an AA and an ES. After CA performs the system initialization, AA aggregates attributes from all data domains to generate a secret key for DU. DO generates ciphertext based on the access policy. When DU wants to decrypt the ciphertext, AA first checks whether DU meets the access policy of the ciphertext. If it meets the access policy, DU can use the secret key to decrypt the ciphertext. When DO wants to update the access authority to the ciphertext, DO uses the new attribute set to generate a new access policy and an update message accordingly.

3.3. Threat Model

In FPCA, CA and AA are trusted, ES is semi-trusted that will honestly execute algorithms, but it may passively collude with adversary A, A can tamper with the access policy of the ciphertext in ES, leading to data leakage or tampering. DO is trusted but DU is semi-trusted, DU performs algorithms honestly but DU can be an adversary that attempts to access the data of other IoT users. Assume that adversary A can obtain any ciphertext in which it is interested but A cannot get a valid secret key that can decrypt the ciphertext correctly. A may perform the following attacks to break FPCA.

1. **Key Forgery Attack:** The adversary A has an invalid secret key that cannot decrypt the ciphertext, A tries to perform the key forgery attack to forge a valid secret key that can decrypt the ciphertext.
2. **Collusion Attack:** There are two possible ways for adversaries to carry out collusion attacks. First, multiple adversaries with no valid secret key may share their invalid secret key with each other and try to generate a valid secret key. Second, because ES is semi-trusted, the adversary A can temper the ciphertext stored in ES by conspiring with ES, so that A can use its invalid secret key to decrypt the tempered ciphertext.
3. **Chosen-Key Attack:** Adversary A can get multiple secret keys with different attribute sets, but none of these can correctly decrypt the ciphertext. A attempts to derive a valid secret key based on the invalid secret keys it acquired to decrypt the ciphertext.
4. **Attack the Updated Ciphertext:** DO revoked the access authority of the adversary A, A tries to decrypt the updated ciphertext based on its old secret key.

4. The Proposed FPCA Scheme

In this section, we first introduce the notation required for FPCA in Table 2. Then, we introduce the multiple data domain model of FPCA, and the scheme definitions of FPCA are defined. Subsequently, we introduce the construction of FPCA.

Table 2. Notations

Parameter	Description
N	Number of data domains
n	Total number of attributes in U_U
At	Attribute
Dm	Data domain
$\hbar$	Attribute flag, indicating whether the attribute At is in the attribute set
CoE	Attribute location public key, utilized by DO to compute the components related to the attributes within the ciphertext.
I_{DU}	The access structure of DU
$f(x, I)$	The attribute polynomial based on access structure I to convert I into element in Z_p to participate in the computation of ciphertext and secret key
s_1, s_2	The secret sharing
dm_1	The auxiliary decryption component

4.1. Multiple Data Domain Model

Here we first introduce the construction of the multiple data domain model. Then, we introduce attribute encoding and attribute aggregation.

The multiple data domain model of FPCA is illustrated in Fig. 2, which can achieve efficient data access across multiple data domains by combining attribute encoding and aggregation. In FPCA, all data domains belong to the same trust domain. Different data domains can manage non-disjoint attribute sets. Each data domain has an AA, AAi from the data domain Dm_i aggregates the attribute sets sent by AA from all other data domains.

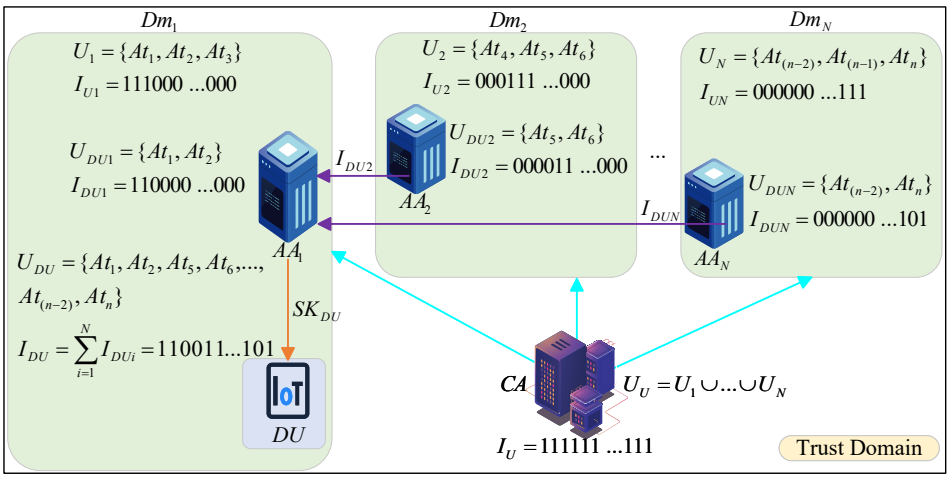


Fig. 2. The Multiple Data Domain Model of FPCA

By encoding attributes, we can quickly aggregate attribute sets from multiple data domains. Assume that U_U is the union of the attribute sets of all data domains, including duplicate attributes, set the attribute set of the data domain Dm_x as $U_x = \{At_i | i \in [1, n]\}$, which is controlled by AA_x, n is the number of attributes in U_U, $U_x \subseteq U_U$. The order of each attribute At_i in U_x and U_U is fixed. We utilize attribute encoding to generate the access structure corresponding to the attribute set. The access structure of U_x is $I_x = \hbar_{x1}..\hbar_{xn}$, which is an n-bit string. If $At_i \in U_x, \hbar_{xi} = 1$, otherwise $\hbar_{xi} = 0$, I_U is the access structure of U_U which is a bit string that entirely consists of 1, $I_U = \sum_{x=1}^{N} I_x$. In this way, the attribute set is converted into a bit string by encoding the attributes as 0 and 1. We select a subset $U_{sj} = \{At_i | i \in [1, n]\}$ for each $U_j, j \in [1, N]$. The access structure of U_{sj} is $I_{sj} = \hbar_{sj1}..\hbar_{sjn}$, if $At_i \in U_{si}, \hbar_{sji} = 1$, otherwise $\hbar_{sji} = 0$. Then, after AA_x in the data domain Dm_x receiving $\{I_{sj} | j \in [1, N], j \neq x\}$ from every other AAs, AA_x performs attribute aggregation through the following operation:

$$Aggregated\ Results = \sum_{j=1}^{N} I_{sj}. \tag{1}$$

In this way, AA_x aggregates all attributes of $\{U_{sj} | j \in [1, N]\}$.

AAs in different data domains can manage a non-disjoint attribute set, and the attribute codes of the attributes in the attribute set managed by each AA correspond to different consecutive bit strings in I_U. Even for the same attributes, their attribute codes are located in different positions in the access structure when they are in different data domains. For At_i and At_j in U_U, they can represent the same attribute when $i \neq j$, At_i and At_j belong to different data domains.

For example, in Fig. 2, CA assigns attribute sets $U_1, ..., U_N$ to $Dm_1, ..., Dm_N$, respectively. $\{I_{Ui} | i \in [1.N]\}$ is the access structure of $\{U_i | i \in [1.N]\}$. DU is in Dm_1. $AA_1, ..., AA_N$ assigns the attribute set $U_{DU1}, ..., U_{DUN}$ to DU, respectively. U_{DU} is the union of $\{U_{DUi} | i \in [1, N]\}$, and U_{DU} is the attribute set of DU. AA_1 in Dm_1 generates I_{DU} by aggregating the access structure $\{I_{DUi} | i \in [1.N]\}$ of $\{U_{DUi} | i \in [1, N]\}$. I_{DU} is the access structure of DU. AA_1 generates the secret key SK_{DU} for DU based on I_{DU}.

4.2. Scheme Definition

The algorithms of the proposed FPCA are defined as follows:

1. **System Initialization** $(1^{\epsilon}) \rightarrow (\boldsymbol{PP}, \boldsymbol{MK})$: CA inputs the security parameter ε, outputs the public parameters PP and the master secret key MK. CA assigns disjoint attribute subsets to each AAi.
2. **MDKG** $(MK, PP) \rightarrow (\boldsymbol{SK_{DU}})$: AA of each data domain defines the access structure I_{DU} for DU together and generates the secret key SK_{DU} for DU based on I_{DU}.
3. **CLCE** $(M, PP) \rightarrow (\boldsymbol{CT})$: DO defines the access policy $\mathfrak{R}$ with the access structure $I_{\mathfrak{R}}$. DO inputs the plaintext M, PP, $I_{\mathfrak{R}}$ and outputs the ciphertext CT.
4. **Decryption** $(CT, ED_{\kappa}, SK_{DU}) \rightarrow (\boldsymbol{M})$: DU gets the ciphertext CT and the encrypted data ED_{κ} from ES. AA checks if DU satisfies the access policy. DU inputs CT, ED_{κ} and the secret key SK_{DU} and outputs the plaintext M.

5. **APU** $(CT, PP, MK) \rightarrow (\boldsymbol{CT_{new}})$: DO inputs the ciphertext CT, PP and MK. DO defines the new access policy $\mathfrak{R}_{new}$, outputs the new ciphertext CT_{new}.

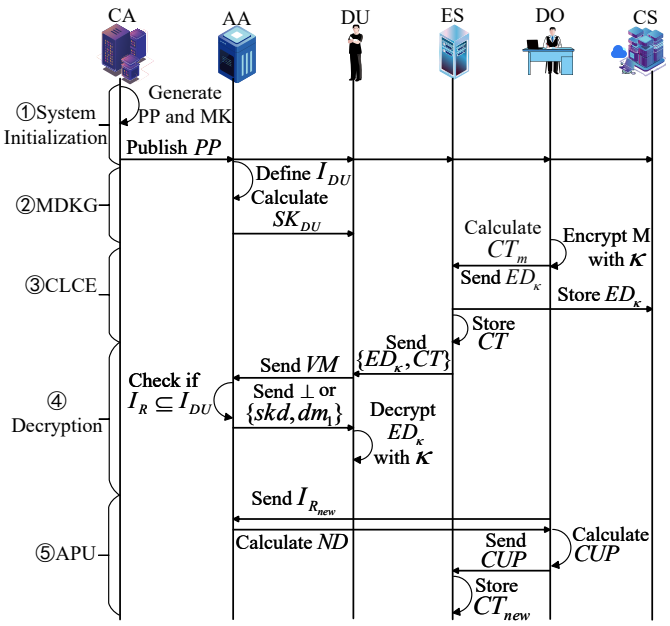


Fig. 3. The Workflow of FPCA

4.3. Construction of FPCA

In this section, we present the detailed construction of the proposed FPCA scheme, made up of five parts: 1) **System Initialization**, 2) **MDKG**, 3) **CLCE**, 4) **Decryption**, and 5) **APU**. The workflow of FPCA is shown in Fig. 3.

1) System Initialization

The System Initialization algorithm is performed by CA and each AA.

- **Step 1**: CA input Security Parameter ε and chooses a bilinear pairing group $BPG = \{G, G^* | G \times G \rightarrow G^*\}$ with prime order p and chooses g as the generator of G. The bilinear pairing is defined as $e(g^x, g^y) = e(g, g)^{xy}$, where $\{x, y \in Z_p\}$.
- **Step 2**: CA defines a hash functions $H : \{0,1\}^* \rightarrow Z_p^*$. H is one-way collision-resistant.
- **Step 3**: CA sets non-disjoint attribute set $\{U_j, j \in [1, N]\}$ for each data domain $Dm_j, j \in [1, N]$, N is the number of data domains. CA defines $U_U = \{At_i | i \in [1, n]\}$, which is the union of the attribute sets of all data domains, $U_j \subseteq U_U$, n is the number of attributes in U_U, and n is invariant. CA assigns $\gamma \in Z_p$ and U_j to AA_j of

each data domain $Dm_j, j \in [1, N]$. We define I_j as the access structure of U_j, I_j is represented as an n-bit string $\{\hbar_1 ... \hbar_n\}$.

$$\begin{cases} \hbar_i = 1, At_i \in U_j, \\ \hbar_i = 0, At_i \notin U_j. \end{cases} \tag{2}$$

- **Step 4**: CA randomly chooses $\{\alpha, q \in Z_p\}$ and sends to each AA. CA computes $CoE = \{coe_i = g^{\gamma^i} | i \in [1, n]\}$ and $BI = \{e(g, g)^\alpha, g^q\}$.
- **Step 5**: CA publishes the public parameters:

$$PP = (BPG, e(\cdot), H, g, p, CoE, BI), \tag{3}$$

and the master secret key is: $MK = (\gamma, q, \alpha)$. For the security of the system, MK is kept by CA and each AA.

2) Multiple Data Domains Key Generation Algorithm

The MDKG algorithm is performed by AA based on the multiple data domain model. The Algorithm 1 shows the details of MDKG.

Algorithm 1: MDKG Algorithm

Input: MK, PP
Output: SK_{DU}
Set n-bit string $I_{DU} = 0...0$;
Set $f(\gamma, I_{DU}) = 1$;
Randomly set $d_{DU_1}, d_{DU_2} \in Z_p$;
for $i \in [1, N]$ **do**
  Generate $I_{DUi} = \hbar_{1DUi} ... \hbar_{nDUi}$;
  Compute $I_{DU} = I_{DU} + I_{DUi}$;
end
for $y \in [1, n]$ **do**
  Compute $f(\gamma, I_{DU}) = f(\gamma, I_{DU}) \cdot (\gamma + H(y))^{1-\hbar_{yDU}}$;
end
Computes
$sk_{DU1} = g^{\alpha + q d_{DU_2}}, sk_{DU2} = g^{d_{DU_2} + d_{DU_1} f(\gamma, I_{DU})}, sk_{DU3} = g^{q d_{DU_1}}$;
Return $SK_{DU} = \{I_{DU}, sk_{DU1}, sk_{DU2}, sk_{DU3}\}$

- **Step 1**: The DU in the data domain Dm_j sends a key generation request to AA_j.
- **Step 2**: AA_j randomly chooses $\{d_{DU_1}, d_{DU_2}\} \in Z_p$ for DU.
- **Step 3**: Each attribute authority $\{AAi | i \in [1, N]\}$ defines an access structure $I_{DUi} = \hbar_{1DUi} ... \hbar_{nDUi}$ for DU based on attributes subset U_{DUi} of DU defined by each AA_i. $\{AA_i | i \in [1, N], i \neq j\}$ sends I_{DUi} to AA_j.
- **Step 4**: AA_j computes:

$$\begin{cases} I_{DU} = \sum_{i=1}^{N} I_{DUi} = \hbar_{1DU}...\hbar_{nDU}, \\ f(\gamma, I_{DU}) = \prod_{y=1}^{n} (\gamma + H(y))^{1-\hbar_{yDU}}, \\ sk_{DU1} = g^{\alpha + qd_{DU_2}}, \\ sk_{DU2} = g^{d_{DU_2} + d_{DU_1} f(\gamma, I_{DU})}, \\ sk_{DU3} = g^{qd_{DU_1}}. \end{cases} \tag{4}$$

AA_j sends the secret key $SK_{DU} = \{I_{DU}, sk_{DU1}, sk_{DU2}, sk_{DU3}\}$ to DU.

SK_{DU} contains only four elements, because n is invariant, the length of I_{DU} is n, which is constant. The attributes assigned to DU by various data domains are aggregated in I_{DU} through the attribute aggregation. The aggregation result I_{DU} is converted into an element in Z_p through $f(\gamma, I_{DU})$ and used to calculate sk_{DU2}. The lengths of sk_{DU1}, sk_{DU2} and sk_{DU3} are $|G|$, $|G|$ is the length of the element in G. Thus, the length of the secret key SK_{DU} is independent of the number of attributes and is constant.

Compared to the standard composite-key ABE, MDKG does not require the secret distribution computation, MDKG can reduce the computational costs and can maintain the length of the secret key constant.

3) Constant-Length Ciphertext Encryption Algorithm

The CLCE algorithm is performed by DO. The Algorithm 2 shows the details of CLCE.

- **Step 1**: The DO of the data domain Dm_j chooses a random number $\kappa \in G^*$ as a symmetric key to encrypt plaintext M with the symmetric encryption algorithm and gets the encrypted data ED_κ. DO sends ED_κ to ES, ES stores ED_κ in CS, CS sends the storage address $Sd(ED_\kappa)$ of ED_κ to ES.
- **Step 2**: DO design an AND-gate access policy $\mathfrak{R}$, $I_{\mathfrak{R}} = \hbar_{1\mathfrak{R}}...\hbar_{n\mathfrak{R}}$ is the access structure of $\mathfrak{R}$, $\mathfrak{R}$ contains only the attributes in U_j, thus the number of "1" in $I_{\mathfrak{R}}$ will not exceed the number of attributes in U_j. Then DO defines:

$$f(x, I_{\mathfrak{R}}) = \prod_{i=1}^{n} (x + H(i))^{1-\hbar_{i\mathfrak{R}}} = e_0 + e_1 x + ... + e_n x^n. \tag{5}$$

  We denote the coefficient of x^i by e_i.
- **Step 3**: DO chooses random numbers $\{s_v, s_1\} \in Z_p$ and sets s_v as the secret value.
- **Step 4**: DO computes:

$$\begin{cases} C_0 = \kappa \cdot e(g, g)^{\alpha s_v}, \\ C_1 = g^{s_v}, \\ C_2 = g^{qs_1}, \\ C_3 = (\prod_{i=0}^{n} coe_i^{e_i})^{s_v} = g^{s_v f(\gamma, I_{\mathfrak{R}})}. \end{cases} \tag{6}$$

Algorithm 2: CLCE algorithm

Input: M, PP
Output: CT
Set $C_3 = 1$;
DO randomly chooses $\kappa \in G^*$;
DO generates ED_κ by encrypts M with κ;
DO sends ED_κ to ES, ES stores ED_κ in CS;
CS returns $Sd(ED_\kappa)$ to ES;
DO generates $I_\mathfrak{R} = \hbar_{1\mathfrak{R}}...\hbar_{n\mathfrak{R}}$;
DO generates $f(x, I_\mathfrak{R}) = \prod_{i=1}^{n}(x + H(i))^{1-\hbar_{i\mathfrak{R}}}$;
DO sets e_i as the coefficient of x^i in $f(x, I_\mathfrak{R})$;
DO randomly chooses $\{s_v, s_1 \in Z_p\}$;
for $i \in [0, n]$ **do**
  DO computes $C_3 = C_3 \cdot (coe_i^{'})^{e_i}$;
end
DO computes $C_3 = C_3^{s_v}$;
DO computes $C_0 = \kappa \cdot e(g, g)^{\alpha s_v}, C_1 = g^{s_v}, C_2 = g^{qs_1}$;
DO sends $CT_m = \{I_\mathfrak{R}, C_0, C_1, C_2, C_3\}$ to ES;
ES stores $CT = \{Sd(ES_\kappa), CT_m\}$;
Return CT

DO stores the middle ciphertext $CT_m = \{I_\mathfrak{R}, C_0, C_1, C_2, C_3\}$ in ES. Finally, ES obtains the complete ciphertext $CT = \{Sd(ED_\kappa), CT_m\}$.

Except for $Sd(ED_\kappa)$, CT contains only five elements, because n is invariant, the length of $I_\mathfrak{R}$ is n, which is constant. As discussed above, $f(\gamma, I_\mathfrak{R})$ is an element in Z_p, the length of C_0 is $|G^*|$, $|G^*|$ is the length of element in G^*. The lengths of C_1, C_2, and C_3 are $|G|$. Thus, the length of the ciphertext CT is independent of the number of attributes and is constant.

4) Decryption

The Decryption algorithm is performed by ES, AA and DU to decrypt CT and ED_κ to obtain plaintext M.

- **Step 1**: The DU in the data domain Dm_j sends an access request to ES, requesting the encrypted data that have attracted its interest. ES finds the matching ciphertext CT and obtains ED_κ from CS according to $Sd(ED_\kappa)$. ES sends ED_κ and CT_m to DU.
- **Step 2**: DU sends the verification message $VM = \{I_\mathfrak{R}, I_{DU}, C_1, C_2, C_3, sk_{DU2}, sk_{DU3}\}$ to AA_j. AA_j checks if the following equation holds:

$$e(C_3, g^{-1})e(g, C_1^{f(\gamma, I_\mathfrak{R})}) \stackrel{?}{=} 1. \tag{7}$$

If Eq. 7 does not hold, it means that CT has been tampered with, if $I_\mathfrak{R} \not\subseteq I_{DU}$, it means that DU does not satisfy the access policy $\mathfrak{R}$ of CT, AA_j return $\perp$ to DU. If Eq. 7 and $I_\mathfrak{R} \subseteq I_{DU}$ hold, AA_j computes the transformed key skd:

$$skd = sk_{DU2} \cdot sk_{DU3}^{(f(\gamma, I_{\mathfrak{R}}) - f(\gamma, I_{DU}))/q}, \tag{8}$$

and AA_j implicitly defines $s_2 = s_v - s_1$ and computes:

$$dm_1 = e(skd, C_1^q / C_2) = e(g, g)^{qs_2 d_{DU_2} + qs_2 d_{DU_1} \cdot f(\gamma, I_{\mathfrak{R}})}. \tag{9}$$

AA_j sends skd and dm_1 to DU.

– **Step 3**: DU computes the intermediate decryption result:

$$dm = \frac{e(skd, C_2) \cdot dm_1}{e(sk_{DU3}, C_3)} = e(g, g)^{qs_v d_{DU_2}}. \tag{10}$$

Then, DU computes:

$$\kappa = \frac{C_0 \cdot dm}{e(C_1, sk_{DU1})}. \tag{11}$$

– **Step 4**: DU decrypt ED_κ with κ to obtain plaintext M.

5) Access Policy Update

The APU algorithm is performed by DO, AA, and ES when DO wants to change the access policy. For example, DO will perform the APU algorithm if DO intends to revoke or add access control privileges for a class of DU. The Algorithm 3 shows the details of the APU algorithm.

Algorithm 3: APU algorithm

Input: CT, PP, MK
Output: CT_{new}
Sets $f(\gamma, I_{\mathfrak{R}_{new}}) = 1$;
DO generates $I_{\mathfrak{R},new} = \hbar_{1\mathfrak{R},new} ... \hbar_{n\mathfrak{R},new}$;
DO sends $I_{\mathfrak{R},new}$ to AA;
for $i \in [1, n]$ **do**
  AA computes $f(\gamma, I_{\mathfrak{R}_{new}}) = f(\gamma, I_{\mathfrak{R}_{new}}) \cdot (\gamma + H(i))^{1 - \hbar_{i\mathfrak{R},new}}$;
end
AA computes $ND = g^{f(\gamma, I_{\mathfrak{R}_{new}})}$ and sends ND to DO;
DO chooses $s_{v,new} \in Z_p$;
DO computes
$C_{0,new} = \kappa \cdot e(g, g)^{\alpha s_{v,new}}, C_{1,new} = g^{s_{v,new}}, C_{3,new} = ND^{s_{v,new}}$;
DO sends $CUP = \{I_{\mathfrak{R}_{new}}, C_{0,new}, C_{1,new}, C_{3,new}\}$ to ES;
ES updates CT_m to $CT_{m,new} = \{CUP, C_2\}$;
Return $CT_{new} = \{Sd(ED_\kappa), CT_{m,new}\}$

– **Step 1**: The DO in the data domain Dm_j sets the new AND-gate access policy $\mathfrak{R}_{new}$, $I_{\mathfrak{R}_{new}} = \hbar_{1\mathfrak{R}_{new}}...\hbar_{n\mathfrak{R}_{new}}$ is the access structure of $\mathfrak{R}_{new}$. DO sends $I_{\mathfrak{R}_{new}}$ to AA_j. AA_j computes:

$$f(\gamma, I_{\mathfrak{R}_{new}}) = \prod_{i=1}^{n} (\gamma + H(i))^{1-\hbar_{i\mathfrak{R}_{new}}}, \tag{12}$$

and sends the attribute update component $ND = g^{f(\gamma, I_{\mathfrak{R}_{new}})}$ to DO.

– **Step 2**: DO chooses a new secret value $s_{v,new} \in Z_p$ and computes:

$$\begin{cases} C_{0,new} = \kappa \cdot e(g,g)^{\alpha s_{v,new}}, \\ C_{1,new} = g^{s_{v,new}}, \\ C_{3,new} = ND^{s_{v,new}} = g^{s_{v,new} f(\gamma, I_{\mathfrak{R}_{new}})}. \end{cases} \tag{13}$$

– **Step 3**: DO sends the update message $CUP = \{I_{\mathfrak{R}_{new}}, C_{0,new}, C_{1,new}, C_{3,new}\}$ to ES. ES obtains the new middle ciphertext $CT_{m,new} = \{CUP, C_2\}$. Finally, ES stores the updated ciphertext $CT_{new} = \{Sd(ED_\kappa), CT_{m,new}\}$.

CUP contains only four elements, because n is invariant, the length of $I_{\mathfrak{R}_{new}}$is n, which is constant. As discussed above, the lengths of $C_{0,new}, C_{1,new}, C_{3,new}$ are the same as in CT. Thus, the length of the update message CUP is independent of the number of attributes and is constant. We only need to update four elements in ciphertext. In this way, APU can significantly reduce the overhead of updating access policies.

4.4. Correctness Analysis

We first analyze the correctness of the decryption. Set the access policy of the ciphertext CT as $\mathfrak{R}$, set the attributes structure of DU as I_{DU}. Let $I_{\mathfrak{R}} \subseteq I_{DU}$. AA checks:

$$\begin{aligned} e(C_3, g^{-1}) e(g, C_1^{f(\gamma, I_{\mathfrak{R}})}) &= e((\prod_{i=0}^{n} coe_i^{e_i})^{s_v}, g^{-1}) e(g, g^{s_v f(\gamma, I_{\mathfrak{R}})}) \\ &= e(g^{s_v f(\gamma, I_{\mathfrak{R}})}, g^{-1}) e(g, g^{r f(\gamma, I_{\mathfrak{R}})}) \\ &= e(g,g)^{-s_v f(\gamma, I_{\mathfrak{R}})} e(g,g)^{s_v f(\gamma, I_{\mathfrak{R}})} \\ &= e(g,g)^{s_v f(\gamma, I_{\mathfrak{R}}) - s_v f(\gamma, I_{\mathfrak{R}})} \\ &= 1. \end{aligned}$$

The Eq. 7 holds, AA computes:

$$\begin{aligned} skd &= sk_{DU2} \cdot sk_{DU3}^{(f(\gamma, I_{\mathfrak{R}}) - f(\gamma, I_{DU}))/q} \\ &= g^{d_{DU_2} + d_{DU_1} f(\gamma, I_{DU})} \cdot g^{q d_{DU_1} \cdot (f(\gamma, I_{\mathfrak{R}}) - f(\gamma, I_{DU}))/q} \\ &= g^{d_{DU_2} + d_{DU_1} f(\gamma, I_{DU})} \cdot g^{d_{DU_1} \cdot f(\gamma, I_{\mathfrak{R}}) - d_{DU_1} \cdot f(\gamma, I_{DU})} \\ &= g^{d_{DU_2} + d_{DU_1} f(\gamma, I_{DU}) + d_{DU_1} \cdot f(\gamma, I_{\mathfrak{R}}) - d_{DU_1} \cdot f(\gamma, I_{DU})} \\ &= g^{d_{DU_2} + d_{DU_1} \cdot f(\gamma, I_{\mathfrak{R}})}, \end{aligned}$$

and

$$\begin{aligned} dm_1 &= e(skd, C_1^q / C_2) \\ &= e(g^{d_{DU_2}+d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}, g^{qs_v}/g^{qs_1}) \\ &= e(g^{d_{DU_2}+d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}, g^{q(s_v-s_1)}) \\ &= e(g,g)^{q(s_v-s_1)d_{DU_2}+q(s_v-s_1)d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})} \\ &= e(g,g)^{qs_2 d_{DU_2}+qs_2 d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}. \end{aligned}$$

Then, DU computes:

$$\begin{aligned} dm &= \frac{e(skd, C_2)\cdot dm_1}{e(sk_{DU3}, C_3)} \\ &= \frac{e(g^{d_{DU_2}+d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}, g^{qs_1})\cdot e(g,g)^{qs_2 d_{DU_2}+qs_2 d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}}{e(g^{qd_{DU_1}}, (\prod_{i=0}^{n} coe_i^{e_i})^{s_v})} \\ &= \frac{e(g,g)^{qs_1 d_{DU_2}+qs_1 d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}\cdot e(g,g)^{qs_2 d_{DU_2}+qs_2 d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}}{e(g^{qd_{DU_1}}, g^{s_v f(\gamma, I_{\mathfrak{R}})})} \\ &= \frac{e(g,g)^{q(s_1+s_2)d_{DU_2}+q(s_1+s_2)d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}}{e(g,g)^{qs_v d_{DU_1} f(\gamma, I_{\mathfrak{R}})}} \\ &= \frac{e(g,g)^{qs_v d_{DU_2}+qs_v d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}}{e(g,g)^{qs_v d_{DU_1} f(\gamma, I_{\mathfrak{R}})}} \\ &= e(g,g)^{qs_v d_{DU_2}}, \end{aligned}$$

and

$$\begin{aligned} \kappa &= \frac{C_0 \cdot dm}{e(C_1, sk_{DU1})} \\ &= \frac{\kappa \cdot e(g,g)^{\alpha s_v}\cdot e(g,g)^{qs_v d_{DU_2}}}{e(g^{s_v}, g^{\alpha+qd_{DU_2}})} \\ &= \frac{\kappa \cdot e(g,g)^{\alpha s_v+qs_v d_{DU_2}}}{e(g,g)^{\alpha s_v+qs_v d_{DU_2}}} \\ &= \kappa. \end{aligned}$$

Thus, DU that satisfy the access policy of the ciphertext can get the symmetric key κ and use it to decrypt ED_κ to get the plaintext M.

Second, we analyze the correctness of the APU algorithm. Assume that DO generates a new access policy $\mathfrak{R}_{new}$ for CT and generates a new secret value $s_{v,new}$. AA computes $ND = g^{f(\gamma, I_{\mathfrak{R}_{new}})}$, DO computes $\{C_{0,new} = \kappa \cdot e(g,g)^{\alpha s_{v,new}}, C_{1,new} = g^{s_{v,new}}, C_{3,new} = ND^{s_{v,new}}\}$. Thus, the new ciphertext $CT_{new} = \{Sd(ED_\kappa), I_{\mathfrak{R}_{new}}, C_{0,new}, C_{1,new}, C_2, C_{3,new}\}$. If DU meets $\mathfrak{R}$ but does not satisfy the new access policy $\mathfrak{R}_{new}$, AA returns $\perp$ to DU. If DU uses SK_{DU} and the old $\{skd, dm_1\}$ to decrypt the

updated ciphertext CT_{new}, then

$$\begin{aligned}
&\frac{e(skd, C_2) \cdot dm_1}{e(sk_{DU3}, C_{3,new})} \\
&= \frac{e(g^{d_{DU_2}+d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}, g^{qs_1}) \cdot e(g,g)^{qs_2 d_{DU_2}+qs_2 d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}}{e(g^{qd_{DU_1}}, ND^{s_{v,new}})} \\
&= \frac{e(g,g)^{qs_1 d_{DU_2}+qs_1 d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})} \cdot e(g,g)^{qs_2 d_{DU_2}+qs_2 d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}}{e(g^{qd_{DU_1}}, g^{s_{v,new} f(\gamma, I_{\mathfrak{R}_{new}})})} \\
&= \frac{e(g,g)^{q(s_1+s_2)d_{DU_2}+q(s_1+s_2)d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})}}{e(g,g)^{qs_{v,new} d_{DU_1} f(\gamma, I_{\mathfrak{R}_{new}})}} \\
&= e(g,g)^{qs_v d_{DU_2}+qs_v d_{DU_1}\cdot f(\gamma, I_{\mathfrak{R}})-qs_{v,new} d_{DU_1} f(\gamma, I_{\mathfrak{R}_{new}})} \neq e(g,g)^{qs_{v,new} d_{DU_2}},
\end{aligned}$$

DU cannot obtain κ. If DU_A satisfies $\mathfrak{R}_{new}$, then $I_{\mathfrak{R}_{new}} \subseteq I_{DU_A}$. Let $SK_{DU_A} = \{I_{DU_A}, sk_{DU_{A,1}} = g^{\alpha+qd_{DU_{A,2}}}, sk_{DU_{A,2}} = g^{d_{DU_{A,2}}+d_{DU_{A,1}} f(\gamma, I_{DU_A})}, sk_{DU_{A,3}} = g^{qd_{DU_{A,1}}}\}$ be the secret key of DU_A, AA implicitly defines $s_{2,new} = s_{v,new} - s_1$ and computes $skd_{new} = sk_{DU_{A,2}} \cdot sk_{DU_{A,3}}^{(f(\gamma, I_{\mathfrak{R}_{new}})-f(\gamma, I_{DU_A}))/q} = g^{d_{DU_{A,2}}+d_{DU_{A,1}}\cdot f(\gamma, I_{\mathfrak{R}_{new}})}, dm_{1,new} = e(skd_{new}, C_{1,new}^q / C_2) = e(g,g)^{qs_{2,new} d_{DU_{A,2}}+qs_{2,new} d_{DU_{A,1}}\cdot f(\gamma, I_{\mathfrak{R}_{new}})}$. DU_A computes

$$\begin{aligned}
dm_{new} &= \frac{e(skd_{new}, C_2) \cdot dm_{1,new}}{e(sk_{DU_{A,3}}, C_{3,new})} \\
&= \frac{e(g^{d_{DU_{A,2}}+d_{DU_{A,1}}\cdot f(\gamma, I_{\mathfrak{R}_{new}})}, g^{qs_1}) \cdot e(g,g)^{qs_{2,new} d_{DU_{A,2}}+qs_{2,new} d_{DU_{A,1}}\cdot f(\gamma, I_{\mathfrak{R}_{new}})}}{e(g^{qd_{DU_{A,1}}}, ND^{s_{v,new}})} \\
&= \frac{e(g,g)^{q(s_1+s_{2,new})d_{DU_{A,2}}+q(s_1+s_{2,new})d_{DU_{A,1}}\cdot f(\gamma, I_{\mathfrak{R}_{new}})} \cdot}{e(g^{qd_{DU_{A,1}}}, g^{s_{v,new} f(\gamma, I_{\mathfrak{R}_{new}})})} \\
&= \frac{e(g,g)^{qs_{v,new} d_{DU_{A,2}}+qs_{v,new} d_{DU_{A,1}}\cdot f(\gamma, I_{\mathfrak{R}_{new}})} \cdot}{e(g,g)^{qs_{v,new} d_{DU_{A,1}} f(\gamma, I_{\mathfrak{R}_{new}})}} \\
&= e(g,g)^{qs_{v,new} d_{DU_{A,2}}},
\end{aligned}$$

and computes

$$\begin{aligned}
&\frac{C_{0,new} \cdot dm_{new}}{e(C_{1,new}, sk_{DU_{A,1}})} \\
&= \frac{\kappa \cdot e(g,g)^{\alpha s_{v,new}} \cdot e(g,g)^{qs_{v,new} d_{DU_{A,2}}}}{e(g^{s_{v,new}}, g^{\alpha+qd_{DU_{A,2}}})} \\
&= \frac{\kappa \cdot e(g,g)^{\alpha s_{v,new}+qs_{v,new} d_{DU_{A,2}}}}{e(g,g)^{\alpha s_{v,new}+qs_{v,new} d_{DU_{A,2}}}} \\
&= \kappa.
\end{aligned}$$

Thus, the APU algorithm can successfully update the ciphertext, the updated ciphertext can be decrypted normally by DU that meet the updated access policy, and can prevent DU that meet the old access policy but do not meet the updated access policy from decrypting the updated ciphertext.

5. Security Analysis

The main goal of the security analysis in our FPCA is to achieve indistinguishability of the data and to resist collusion attacks on the secret key and the ciphertext. We combined the Discrete Logarithm (DL) problem, the Decisional Diffie-Hellman (DDH) problem, the Decisional Bilinear Diffie-Hellman (DBDH) problem, and the selective game for ABE for security analysis. The hash function H is a random oracle. In security analysis, adversaries do not have valid secret keys to decrypt the challenge ciphertext or the updated ciphertext. The adversary can apply for multiple invalid keys and attempt to synthesize valid keys using these invalid keys. Adversaries can share their invalid secret keys with other adversaries and freely obtain the ciphertext from ES. As ES is semi-trusted, the adversary can also conspire with ES.

5.1. Formal Security Analysis

Theorem 1: Based on the DBDH problem, if an adversary A can break FPCA in probabilistic polynomial time (PPT) with an advantage Ω that is non-negligible, then a PPT simulator $\mathfrak{B}$ can be constructed to break the DBDH problem with an advantage $\Omega/2$.

Proof: The challenger CH is trusted, CH can correctly execute FPCA, CH is responsible for responding to adversary A's query and generating challenge ciphertext, the purpose of CH is to maintain the security of the selective game. Given the bilinear pairing group $\{p, g, G, G^*\}$, CH randomly chooses $\{a, b, c \in Z_p\}$ and generates a DBDH instance $DI = \{g, g^a, g^b, g^c, Z\}$, CH randomly chooses $\mu \in \{0, 1\}$, if $\mu = 0$, then $Z = e(g, g)^{abc}$, otherwise, Z is a random number in G^*. CH sends DI to the simulator $\mathfrak{B}$. $\mathfrak{B}$ can only use the DBDH instance to simulate FPCA. $\mathfrak{B}$ plays the role of CH in the interaction with A, but $\mathfrak{B}$ cannot obtain $\{a, b, c\}$. $\mathfrak{B}$ must be able to respond to all valid key queries, and the keys to which it responds must match the real key distribution. The challenge ciphertext generated by $\mathfrak{B}$ must be indistinguishable from the real encryption. The purpose of $\mathfrak{B}$ is to solve the DBDH problem. A submits two messages of equal length and challenge access policy, A can continuously request secret keys from $\mathfrak{B}$, but none of these secret keys can satisfy the challenge access policy, the purpose of A is to break the security of FPCA.

1. ***Init***: A submits a challenge access policy $\mathfrak{R}1$ to $\mathfrak{B}$, $I_{\mathfrak{R}1} = \hbar_{1\mathfrak{R}1}...\hbar_{n\mathfrak{R}1}$ is the access structure of $\mathfrak{R}1$.
2. ***Setup***: $\mathfrak{B}$ randomly chooses $\{\alpha_1, \gamma_c\} \in Z_p$ and computes $CoE_c = \{coe_{ci} = g^{\gamma_c^i} | i \in [1, n]\}$. $\mathfrak{B}$ computes $Y = e(g, g)^{\alpha_1} e(g, g)^{ab} = e(g, g)^{\alpha_1 + ab} = e(g, g)^{\alpha}$. $\mathfrak{B}$ sets $g^{q_c} = g^b$ and sets $BI_c = \{Y, g^{q_c}\}$ and hash function $H : \{0, 1\}^* \to Z_p^*$. $\mathfrak{B}$ outputs the public parameters to A:

$$PP = (BPG, e(\cdot), H, g, p, CoE_c, BI_c).$$

3. ***Key Query 1***: After A sends the key request and an attributes set U_A to $\mathfrak{B}$, $\mathfrak{R}1 \not\subseteq U_A$, $I_A = \hbar_{1A}...\hbar_{nA}$ is the access structure of U_A. $\mathfrak{B}$ randomly chooses $d_{c1} \in Z_p$ and

chooses $d_{c2}^{'} \in Z_p$, then $\mathfrak{B}$ implicitly defines $d_{c2} = d_{c2}^{'} - a$. $\mathfrak{B}$ computes

$$\begin{cases} sk_{s1} = g^{\alpha_1 + bd_{c2}^{'}} = g^{\alpha_1 + ab + b(d_{c2}^{'} - a)} = g^{\alpha + q_c d_{c2}}, \\ sk_{s2} = \dfrac{g^{d_{c2}^{'}}}{g^a} \cdot g^{d_{c1} f(\gamma_c, I_A)} = g^{d_{c2} + d_{c1} f(\gamma_c, I_A)}, \\ sk_{s3} = g^{bd_{c1}} = g^{q_c d_{c1}}. \end{cases}$$

$\mathfrak{B}$ sends the simulated secret key $SK_A = \{I_A, sk_{s1}, sk_{s2}, sk_{s3}\}$ to A, $\mathfrak{R}\mathfrak{1}$ cannot be satisfied by any secret key requested by A from $\mathfrak{B}$.
In the real secret key, $\{d_{A1}, d_{A2}\} \in_R Z_p$ are chosen by AA, $\{sk_1 = g^{\alpha + q_c d_{A1}}, sk_2 = g^{d_{A2} + d_{A1} f(\gamma_c, I_A)}, sk_3 = g^{q_c d_{A1}}\} \in G$. In SK_A, because $\{d_{c1}, d_{c2}\} \in_R Z_p$, $\{sk_{s1}, sk_{s2}, sk_{s3}\} \in G$ and the distributions of $\{d_{A1}, d_{A2}\}$ and $\{d_{c1}, d_{c2}\}$ in Z_p are identical, therefore, the distributions of $\{sk_1, sk_2, sk_3\}$ and $\{sk_{s1}, sk_{s2}, sk_{s3}\}$ in G are identical. A does not know $\{\alpha, \gamma_c, q_c, d_{c1}, d_{c2}\}$, therefore A cannot distinguish between the simulated secret key SK_A and the real secret key, in the view of A, SK_A is real.

4. ***Challenge***: A sends two plaintexts $\{M_0, M_1\}$ to $\mathfrak{B}$ as the challenge query, the lengths of $\{M_0, M_1\}$ are the same. $\mathfrak{B}$ randomly chooses $\beta \in \{0, 1\}$ and $s_1 \in Z_p$. $\mathfrak{B}$ sets $g^{s_c} = g^c$, s_c is the secret value. $\mathfrak{B}$ computes

$$\begin{cases} C_{c0} = M_\beta \cdot Z \cdot e(g, g)^{\alpha_1 c}, \\ C_{c1} = g^c = g^{s_c}, \\ C_{c2} = g^{bs_1} = g^{q_c s_1}, \\ C_{c3} = g^{cf(\gamma_c, I_{\mathfrak{R}\mathfrak{1}})} = g^{s_c f(\gamma_c, I_{\mathfrak{R}\mathfrak{1}})}, \end{cases}$$

and sends the challenge ciphertext $CT_c = \{I_{\mathfrak{R}\mathfrak{1}}, C_{c0}, C_{c1}, C_{c2}, C_{c3}\}$ to A.
In addition, $\mathfrak{B}$ randomly chooses $s_2 \in Z_p$ and computes

$$\begin{aligned} dm_{1c} &= e(\frac{g^{d_{c2}^{'}}}{g^a} \cdot g^{d_{c1} f(\gamma_c, I_{\mathfrak{R}\mathfrak{1}})}, g^b)^{s_2} \\ &= e(g^{d_{c2} + d_{c1} f(\gamma_c, I_{\mathfrak{R}\mathfrak{1}})}, g^b)^{s_2} \\ &= e(g, g)^{bs_2 d_{c2} + bs_2 d_{c1} f(\gamma_c, I_{\mathfrak{R}\mathfrak{1}})} \\ &= e(g, g)^{q_c s_2 d_{c2} + q_c s_2 d_{c1} f(\gamma_c, I_{\mathfrak{R}\mathfrak{1}})}, \end{aligned}$$

$\mathfrak{B}$ sends dm_{1c} to A.
In real ciphertext, $\{s_v, s_1\} \in_R Z_p$, $C_0 = M_\beta \cdot e(g, g)^{\alpha s_v} \in G^*$ and $\{C_1 = g^{s_v}, C_2 = g^{q_c s_v}, C_3 = g^{s_v f(\gamma, I_{\mathfrak{R}\mathfrak{1}})}\} \in G$, in CT_c, $\{s_c, s_1\} \in_R Z_p$, $\{Z, C_{c0}\} \in G^*$ and $\{C_{c1}, C_{c2}, C_{c3}\} \in G$, because the distributions of $\{s_v, s_1\}$ in real ciphertext and $\{s_c, s_1\}$ in CT_c are identical in Z_p, thus, the distributions of $\{C_1, C_2, C_3\}$ and $\{C_{c1}, C_{c2}, C_{c3}\}$ in G are identical. The distribution of $Z \cdot e(g, g)^{\alpha_1 c}$ and $e(g, g)^{\alpha s_v}$ is identical in G^*, then the distribution of C_0 and C_{c0} is identical in G^*.
The real auxiliary decryption component $dm_1 = e(g, g)^{q_c s_2 d_{A_2} + q_c s_2 d_{A_1} \cdot f(\gamma_c, I_{\mathfrak{R}\mathfrak{1}})}$, as discussed above, $\{s_2, d_{A_1}, d_{A_2}, d_{c1}, d_{c2}\} \in_R Z_p$, $\{dm_1, dm_{1c}\} \in G$, the distributions of dm_1 and dm_{1c} are identical in G. $\{s_c, s_1, s_2, \alpha, \gamma_c, q_c, d_{c1}, d_{c2}, Z\}$ are not

disclosed to A and because $\mathfrak{R}1 \not\subseteq U_A$, A cannot decrypt CT_c correctly by using dm_{1c} and A cannot determine whether $s_1 + s_2$ equals S_c. Therefore, from the view of A, CT_c and dm_{1c} are both real and not simulated.

5. ***Update Ciphertext***: A generates a new challenge access policy $\mathfrak{R}2$, $I_{\mathfrak{R}2} = \hbar_{1\mathfrak{R}2}...\hbar_{n\mathfrak{R}2}$ is the access structure of $\mathfrak{R}2$, $\mathfrak{R}2 \not\subseteq U_A$, $\mathfrak{R}2$ cannot be satisfied by any secret key that A requests from $\mathfrak{B}$. A submits $\{CT_c, \mathfrak{R}2\}$ to $\mathfrak{B}$ as the update ciphertext query. $\mathfrak{B}$ randomly chooses $s_n \in Z_p$ and computes $g^{s_{c,new}} = g^{cs_n}$, $s_{c,new}$ is the new secret value. $\mathfrak{B}$ computes

$$\begin{cases} C_{c0,new} = M_\beta \cdot (Z \cdot e(g,g)^{\alpha_1 c})^{s_n}, \\ C_{c1,new} = g^{cs_n} = g^{s_{c,new}}, \\ C_{c3,new} = g^{cs_n f(\gamma_c, I_{\mathfrak{R}2})} = g^{s_{c,new} f(\gamma_c, I_{\mathfrak{R}2})}, \end{cases}$$

and sends the updated challenge ciphertext $CT_{c,new} = \{I_{\mathfrak{R}2}, C_{c0,new}, C_{c1,new}, C_{c2}, C_{c3,new}\}$ to A.
In real updated ciphertext, $s_{v,new} \in_R Z_p$, $C_{0,new} \in G^*$, $\{C_{1,new}, C_{3,new}\} \in G$, in the updated challenge ciphertext, $Z \in G^*$, $s_n \in_R Z_p$, and $c = s_c \in_R Z_p$, then $s_{c,new} = cs_n \in_R Z_p$, $C_{c0,new} \in G^*$, $\{C_{c1,new}, C_{c3,new}\} \in G$. Because A does not obtain $\{s_c, s_n, \gamma_c, \alpha_1, Z\}$, the distributions of $\{C_{c1,new}, C_{c3,new}\}$ and $\{C_{1,new}, C_{3,new}\}$ are identical in G, and the distribution of $\{C_{c0,new}\}$ and $\{C_{0,new}\}$ are identical in G^*, thus, in the view of A, $CT_{c,new}$ is real.
6. ***Key Query 2***: Repeat **Key Query 1**.
7. ***Guess***: A outputs $\beta_1 \in \{0,1\}$ as a guess of β, $\mathfrak{B}$ outputs $\mu_1 \in \{0,1\}$ as a guess of μ. If $\beta_1 = \beta$, then $\mathfrak{B}$ outputs $\mu_1 = 0$ to guess $Z = e(g,g)^{abc}$, otherwise, $\mathfrak{B}$ outputs $\mu_1 = 1$ to guess Z is a random number in G^*.

If $\mu = 0$, then $C_{c0} = M_\beta \cdot e(g,g)^{abc+\alpha_1 c} = M_\beta \cdot e(g,g)^{\alpha s_c}$ and $C_{c0,new} = M_\beta \cdot e(g,g)^{(abc+\alpha_1 c)s_n} = M_\beta \cdot e(g,g)^{\alpha s_{c,new}}$ are valid ciphertext components. A has the advantage Ω of guessing β correctly. In this case, the probability of $\mathfrak{B}$ guessing correctly is $Adv^0_{\mathfrak{B}} = P[\beta_1 = \beta | \mu = 0] = P[\mu_1 = \mu | \mu = 0] = \Omega + 1/2$.

If $\mu = 1$, then Z is a random number in G^*. Thus, C_{c0} and $C_{c0,new}$ are random numbers in G^* in the view of A. A has no advantage in guessing β correctly. In this case, the probability of $\mathfrak{B}$ guessing correctly is $Adv^1_{\mathfrak{B}} = P[\beta_1 \neq \beta | \mu = 1] = P[\mu_1 = \mu | \mu = 1] = 1/2$.

The advantage of $\mathfrak{B}$ in breaking the DBDH problem is

$$\begin{aligned} Adv^{DBDH}_{\mathfrak{B}} &= (Adv^1_{\mathfrak{B}} + Adv^0_{\mathfrak{B}} - 1)/2 \\ &= (1/2 + \Omega + 1/2 - 1)/2 \\ &= \Omega/2. \end{aligned}$$

Thus, the **Theorem 1** is proved.

5.2. Informal Security Analysis

1) Resist Key Forgery Attack:

Assume that there is a ciphertext $CT_{I_{\mathfrak{R}}}$ under the access policy $\mathfrak{R}$, $I_{\mathfrak{R}}$ is the access structure of $\mathfrak{R}$. The secret key of A is $SK_A = \{sk_{A1} = g^{\alpha + qd_{A2}}, sk_{A2} = g^{d_{A2} + d_{A1} f(\gamma, I_A)}$,

$sk_{A3} = g^{qd_{A1}}\}$, $I_{\mathfrak{R}} \not\subseteq I_A$. A attempts to decrypt $CT_{I_{\mathfrak{R}}}$ by forging a valid secret key that meets the access policy $\mathfrak{R}$. To achieve this, A must calculate $sk_{\mathfrak{R},2} = g^{d_{A2}+d_{A1}f(\gamma,I_{\mathfrak{R}})}$. Because, $\{\gamma, q, d_{A1}, d_{A2}\}$ are unknown to A. A cannot calculate $f(\gamma, I_{\mathfrak{R}})$ without γ and cannot calculate $\{g^{d_{A1}}, g^{d_{A2}}\}$ without q. According to the DDH problem, even if A can obtain $g^{d_{A1}f(\gamma,I_{\mathfrak{R}})}$ and calculate $g^{f(\gamma,I_{\mathfrak{R}})}$, but A cannot calculate $g^{d_{A1}f(\gamma,I_{\mathfrak{R}})}$ by $g^{d_{A1}}$ and $g^{f(\gamma,I_{\mathfrak{R}})}$. Thus, A cannot calculate $sk_{\mathfrak{R},2} = g^{d_{A2}+d_{A1}f(\gamma,I_{\mathfrak{R}})}$, our FPCA can resist key forgery attacks.

2) Resist Collusion Attack:

Between multiple adversaries: Assume that adversaries A_1 and A_2 have secret keys $SK_{A1} = \{sk_{A11} = g^{\alpha+qd_{A12}}, sk_{A12} = g^{d_{A12}+d_{A11}f(\gamma,I_{A1})}, sk_{A13} = g^{qd_{A11}}\}$ and $SK_{A2} = \{sk_{A21} = g^{\alpha+qd_{A22}}, sk_{A22} = g^{d_{A22}+d_{A21}f(\gamma,I_{A2})}, sk_{A23} = g^{qd_{A21}}\}$, respectively. Both I_{A_1} and I_{A_2} do not meet the access policy $\mathfrak{R}$ of the ciphertext $CT_{I_{\mathfrak{R}}}$, but $I_{\mathfrak{R}} \subseteq I_{A_1} + I_{A_2}$. A_1 and A_2 attempted to derive $sk_{A12,col} = g^{d_{A12}+d_{A11}f(\gamma,I_{A_1}+I_{A_2})}$ to replace sk_{A12} to decrypt $CT_{I_{\mathfrak{R}}}$ by collusion attack. To achieve this, A_1 and A_2 must obtain d_{A11}, $g^{d_{A11}}$ or $f(\gamma, I_{A_1} + I_{A_2})$. But $\{\gamma, q, d_{A1}, d_{A2}\}$ are unknown to A_1 and A_2. According to the analysis of resisting key forgery attack, the DL problem and the DDH problem, A_1 and A_2 cannot obtain d_{A11}, $g^{d_{A11}}$ or $f(\gamma, I_{A_1} + I_{A_2})$ and cannot calculate $g^{d_{A12}+d_{A11}f(\gamma,I_{A_1}+I_{A_2})}$. Thus, multiple adversaries cannot forge a valid secret key to decrypt the ciphertext by collusion attack.

Between ES and adversary: Because ES is semi-trusted, adversary A may temper with the ciphertext stored in ES by conspiring with ES. Assume that ciphertext $CT_{I_{\mathfrak{R}}} = \{I_{\mathfrak{R}}, C_0 = \kappa \cdot e(g,g)^{\alpha s_v}, C_1 = g^{s_v}, C_2 = g^{qs_1}, C_3 = g^{s_v f(\gamma,I_{\mathfrak{R}})}\}$, the secret key of A is $SK_A = \{sk_{A1} = g^{\alpha+qd_{A2}}, sk_{A2} = g^{d_{A2}+d_{A1}f(\gamma,I_A)}, sk_{A3} = g^{qd_{A1}}\}$, $I_{\mathfrak{R}} \not\subseteq I_A$. A attempts to replace $\{I_{\mathfrak{R}}, C_3\}$ by $\{I_A, C_{A3} = g^{s_v f(\gamma,I_A)}\}$, so that A can decrypt $CT_{I_{\mathfrak{R}}}$ by SK_A. A has g^{s_v} and $g^{f(\gamma,I_A)}$, but γ and s_v are unknown to A, A cannot calculate $f(\gamma, I_A)$. According to the DDH problem, A cannot calculate $C_{A3} = g^{s_v f(\gamma,I_A)}$ by g^{s_v} and $g^{f(\gamma,I_A)}$ and according to the DL problem A cannot calculate $\{f(\gamma, I_A), s_v\}$ by and $g^{f(\gamma,I_A)}$ and g^{s_v}. According to Eq. 7, $e(C_3, g^{-1})e(g, C_1^{f(\gamma,I_A)}) \neq 1$, AA will return $\perp$ to A. Thus, A cannot decrypt $CT_{I_{\mathfrak{R}}}$. Thus, the adversary cannot forge a valid ciphertext by conspiring with ES.

3) Resist Chosen-Key Attack

Assume that adversaries A obtained multiple secret keys with different attribute set: $\{SK_{Ai} = \{sk_{A1} = g^{\alpha+qd_{A2}}, sk_{A2} = g^{d_{A2}+d_{A1}f(\gamma,I_{Ai})}, sk_{A3} = g^{qd_{A1}}\}|i \in [1, L]\}$, L is the number of secret keys that A obtained. None of $\{I_{Ai}|i \in [1, L]\}$ meets the access policy $\mathfrak{R}$ of ciphertext $CT_{I_{\mathfrak{R}}}$, but $I_{\mathfrak{R}} \subseteq \sum_{i=1}^{L} I_{Ai}$. Thus, A attempts to derive a valid secret key by calculating $sk_{A3,val} = g^{d_{A2}+d_{A1}f(\gamma,\sum_{i=1}^{L} I_{Ai})}$ to decrypt $CT_{I_{\mathfrak{R}}}$. To obtain $sk_{A3,val}$, A first needs to calculate $g^{d_{A1}f(\gamma,\sum_{i=1}^{L} I_{Ai})}$. A has $g^{qd_{A1}}$ and $g^{f(\gamma,\sum_{i=1}^{L} I_{Ai})}$, but $\{\gamma, q, d_{A1}, d_{A2}, f(\gamma, \sum_{i=1}^{L} I_{Ai})\}$ are unknown to A. According to the DL problem, A cannot calculate $\{q, d_{A1}, f(\gamma, \sum_{i=1}^{L} I_{Ai})\}$ by g^q, $g^{qd_{A1}}$ and $g^{f(\gamma,\sum_{i=1}^{L} I_{Ai})}$, and according to the DDH problem, A cannot calculate $g^{qd_{A1}f(\gamma,\sum_{i=1}^{L} I_{Ai})}$ by $g^{qd_{A1}}$ and $g^{f(\gamma,\sum_{i=1}^{L} I_{Ai})}$, and cannot calculate $g^{d_{A1}f(\gamma,\sum_{i=1}^{L} I_{Ai})}$. Thus, A cannot obtain $sk_{A3,val}$, our FPCA can resist the chosen-key attack.

4) FPCA Satisfies Backward Security

Assume DO wants to revoke the access permission of A to the ciphertext $CT_{I_A} = \{I_A, C_0 = \kappa \cdot e(g,g)^{\alpha s_o}, C_1 = g^{s_o}, C_2 = g^{qs_1}, C_3 = g^{s_o f(\gamma, I_A)}\}$, DO changes κ to $\kappa 1$ and performs the APU algorithm to update the ciphertext CT_{I_A} to $CT_{I_\Re} = \{I_\Re, C_0 = \kappa 1 \cdot e(g,g)^{\alpha s_v}, C_1 = g^{s_v}, C_2 = g^{qs_1}, C_3 = g^{s_v f(\gamma, I_\Re)}\}$, $I_\Re \not\subseteq I_A$, A cannot use the secret key $SK_A = \{sk_{A1} = g^{\alpha + qd_{A2}}, sk_{A2} = g^{d_{A2} + d_{A1} f(\gamma, I_A)}, sk_{A3} = g^{qd_{A1}}\}$ to decrypt $CT_{I_\Re}$. From the above analysis, we can see that A cannot calculate $sk_{A2,new} = g^{d_{A2} + d_{A1} f(\gamma, I_\Re)}$ or $C_{3,new} = g^{s_v f(\gamma, I_A)}$ for decryption. To obtain $\kappa 1$, A has to calculate $e(g,g)^{\alpha s_v}$. A has $e(g,g)^{\alpha}$ and g^{s_v}, but α and s_v are unknown to A. Thus, according to the DBDH problem, A cannot calculate $e(g,g)^{\alpha s_v}$ by $e(g,g)^{\alpha}$ and g^{s_v}. A does not know g^{α}, then A cannot calculate $e(g,g)^{\alpha s_v}$ by the bilinear pairing operation. Thus, A cannot obtain $e(g,g)^{\alpha s_v}$. In addition, A may try to calculate $\kappa 1 \cdot (e(g,g)^{\alpha s_v})^{s_o / s_v} = \kappa 1 \cdot e(g,g)^{\alpha s_o}$ and use $e(g,g)^{\alpha s_o}$ to obtain $\kappa 1$. But, according to the DL problem, A cannot calculate $\{s_o, s_v\}$ by $\{g^{s_o}, g^{s_v}\}$ and A does not have $e(g,g)^{\alpha s_v}$. Thus, A cannot calculate $\kappa 1 \cdot e(g,g)^{\alpha s_o}$ to get $\kappa 1$, A cannot decrypt the updated ciphertext by using the old secret key, our FPCA satisfies the backward security.

6. Performance Analysis and Evaluation

6.1. Experiment Setting

We perform simulation experiments for the performance evaluation of FPCA on storage, communication, and computation costs. We choose Sun *et al.* [20], Fan *et al.* [6], Li *et al.* [10], and Wu *et al.* [2] as comparison schemes. We implemented servers including CA, AA, ES, and CS under the Win 11 64-bit operating system with Intel(R)core(TM) i3-10105 3.70GHz, 12GB memory. We deploy the IoT network on Win 11 64-bit operating system with 2.5 GHz AMD A10-9620P RADEON R5 and 8GB of memory. We perform the simulation experiment under the jdk1.8.0-152 environment with the JPBC library.

In this experiment, we set the Type-A curve $E(F^v) : y^2 = x^3 + x$ with an order p of 160 $bits$ and $|v| = 512$ $bits$ for [20], [6], [10], [2] and FPCA. We choose G and G^* from the subgroups of $E(F^v)$. As the lengths of $|G|$ and $|G^*|$ are 1024 $bits$, we use $|G|$ to represent the length of the element in G or G^*. Table 3 shows the parameters required for the experiment.

For the data access across data domain simulation experiment, we set that there are two data domains in the system, $NDD = 2$, DU in the data domain Dm_1 access data from the data domain Dm_2, both Dm_1 and Dm_2 have 50 attributes, $n = 100$, and $NA_{C1} = NA_{C2} = 20$.

6.2. Theoretical Analysis

Regarding the theoretical analysis for FPCA with other comparison schemes, we first perform the feature analysis in Table 1. [20], [6], [10], and FPCA support cross data domain access control. Only FPCA and [2] can achieve constant ciphertext length for better access control efficiency, but [2] cannot achieve constant secret key length, while FPCA maintains the secret key length constant. FPCA, [6], and [10] can update the access policy. Only FPCA can maintain the update message in a constant length. Thus, we choose

Table 3. Experiment Parameters

Parameter	Description
NA	Number of attributes
NA_{DU}	Number of attributes for the DU
NA_T	Number of attributes in the access policy
NA_{msa}	Minimum number of attributes that satisfy the access policy for DU
NA_{add}	Number of attributes added to access policy
NA_{rev}	Number of attributes revoked from access policy
NDD	Number of data domains
NA_{Ui}	Number of attributes of DU in data domain Dm_i
T_G	The running time of multiplication operations in group
T_{EXP}	The running time of modular exponentiation operation
T_{BP}	The running time of bilinear pairing

[20], [6], and [10] as compared schemes for cross data domain access control, we choose [6] and [10] as compared schemes for access policy updates, and we choose [2] as a compared scheme for constant-length ABE.

We have summarized the complexity of FPCA and other compared schemes in Table 6, where KG denotes key generation, EN is encryption, DE is decryption, $NA_B = NA_{DU} + NA_T$, $NA_E = NA_T + NA_T + NA_{add} - NA_{rev}$, $NA_C = NA_T + NA_{msa}$.

Table 4 shows the storage costs. Public parameters PP are stored by almost all entities in the access control system. The length of PP in FPCA is less than in other compared schemes. In addition, the ciphertext is stored by DO, DU and ES, the secret key is stored by DU and AA, and the update message is stored by DO and ES. AA in FPCA also stores VM, which is the verification message of FPCA, the storage costs of VM are $2n + 5|G|$. According to Table 6, since n is invariant, the storage complexity of the secret key, the ciphertext and the update message for FPCA are $O(1)$, while for [20], [6] and [10] are $O(NA_{DU})$, $O(NA_T)$ and $O(NA_E - NA_T)$, respectively. For [2], the storage complexity of the ciphertext is $O(1)$, but the storage complexity of the secret key is $O(NA_{DU})$, which is higher than FPCA. The length of the ciphertext in [2] is $4|G| - n$ longer than in FPCA. Therefore, the storage costs of FPCA are lower than those of other compared schemes.

In addition, we also present the communication costs in Table 4. In the system initialization phase, CA sends the public parameters PP to all other entities. In the key generation phase, AA sends the secret key to DU. In the encryption phase, DO sends the middle ciphertext to ES, and DO sends the verification message to ES in [2]. In the decryption phase, ES sends the ciphertext to DU. In FPCA, VM, skd, and dm_1 are also transmitted between AA and DU when decryption. In the access policy update phase, DO sends the update message to ES, and in FPCA, ND and the access structure $I_{\mathfrak{R}_{new}}$ of the new access policy $\mathfrak{R}_{new}$ are also transmitted between AA and DO. The lengths of messages transmitted in FPCA are constant. Thus, according to Table 6, the communication complexity in FPCA is $O(1)$ and is lower than [20], [6], and [10]. Although the communication complexity of the ciphertext in [2] is $O(1)$, the sum of the lengths of the

Table 4. The Comparison of Storage and Communication Costs

Scheme	Public Parameters Length	Secret Key Length	Ciphertext Length	Update Message Length
[20]	$\|Z_p\| + 3\|G\| + (7+3n)\|G\|$	$8\|G\| + 3\|G\| \cdot NA_{DU}$	$4\|G\| + 2\|G\| \cdot NA_T$	/
[6]	$(\|Z_p\| + \|G\|) \cdot (1+n) + 3\|G\|$	$8\|G\| \cdot NA_{DU}$	$4\|G\| + (5\|G\| + 4\|Z_p\|) \cdot NA_T$	$\|G\| + 4\|Z_p\| \cdot (NA_T - NA_{rev}) + (2\|G\| + 3\|Z_p\|) \cdot NA_{add}$
[10]	$6\|G\| + (\|Z_p\| + 2\|G\|) \cdot n$	$(NA_{DU}+2) \cdot 3\|G\|$	$2\|G\| + 5\|G\| \cdot NA_T + \|Z_p\|$	$4\|G\| \cdot NA_T + 2\|G\| \cdot (NA_T - NA_{rev} + NA_{add})$
[2]	$6\|G\| + (\|Z_p\| + \|G\|) \cdot n$	$2\|G\| + 2NA_{DU} \cdot (\|G\| + \|Z_p\|)$	$8\|G\|$	/
FPCA	$\|Z_p\| + 3\|G\| + n \cdot \|G\|$	$n + 3\|G\|$	$n + 4\|G\|$	$n + 3\|G\|$

Table 5. The Comparison of Computational Costs

Scheme	Key Generation	Encryption	Decryption	Access Policy Update
[20]	$4T_{EXP} + T_{BP} + (6T_{EXP} + 3T_G) \cdot (NA_{DU}+1)$	$4T_{EXP} + (8T_{EXP} + 5T_G) \cdot (NA_T + 1)$	$6T_{BP} + 4T_{EXP} + 4T_G + (3T_{BP} + T_{EXP} + 2T_G) \cdot NA_{msa}$	/
[6]	$(12T_{EXP} + 3T_G) \cdot NA_{DU} + T_{EXP}$	$T_{BP} + 4T_{EXP} + T_G + (8T_{EXP} + 2T_{BP} + T_G) \cdot NA_T$	$NA_T \cdot (3T_{EXP} + 5T_G) + 5T_{BP} - 4T_G + NA_{msa} \cdot (5T_{EXP} + 3T_{BP} + 6T_G)$	$T_{EXP} + T_G + (NA_T - NA_{rev} + NA_{add}) \cdot (5T_{EXP} + 3T_G + T_{BP})$
[10]	$11T_{EXP} + T_G + NA_{DU} \cdot (3T_{EXP} + T_G)$	$3T_{EXP} + 2T_G + (1 + NA_T) \cdot (4T_{EXP} + T_G) + 3 \cdot T_{BP}$	$NA_{msa} \cdot (4T_{EXP} + 3T_{BP} + 3T_G) + 2T_{BP} + 4T_{EXP} + T_G$	$4T_{EXP} + T_{BP} + T_G + 2(NA_T - NA_{rev} + NA_{add}) \cdot (T_{EXP} + T_G)$
[2]	$3T_{EXP} \cdot (1 + NA_{DU}) + T_G \cdot (2NA_{DU} + 1)$	$(2T_{EXP} + 2T_G + T_{BP}) \cdot NA_T + T_{EXP}$	$2T_{BP} + NA_{msa} \cdot (2T_{BP} + T_G)$	/
FPCA	$3T_{EXP}$	$4T_{EXP} + NA_T \cdot (T_{EXP} + T_G) + T_G$	$6T_{BP} + 4T_G + 3T_{EXP}$	$4T_{EXP} + 2T_G$

Table 6. The Comparison of Complexity

Scheme	Storage Complexity					Communication Complexity			Computational Complexity			
	CA	AA	DU	DO	ES	AA & DU	DO & ES	DU & ES	KG	EN	DE	AUP
[20]	$O(1)$	$O(NA_{DU})$	$O(NA_B)$	$O(NA_T)$	$O(NA_T)$	$O(NA_{DU})$	$O(NA_T)$	$O(NA_T)$	$O(NA_{DU})$	$O(NA_T)$	$O(NA_{msa})$	/
[6]	$O(1)$	$O(NA_{DU})$	$O(NA_B)$	$O(NA_T)$	$O(NA_E)$	$O(NA_{DU})$	$O(NA_E)$	$O(NA_T)$	$O(NA_{DU})$	$O(NA_T)$	$O(NA_C)$	$O(NA_E)$
[10]	$O(1)$	$O(NA_{DU})$	$O(NA_B)$	$O(NA_T)$	$O(NA_E)$	$O(NA_{DU})$	$O(NA_E)$	$O(NA_T)$	$O(NA_{DU})$	$O(NA_T)$	$O(NA_{msa})$	$O(NA_E)$
[2]	$O(1)$	$O(NA_{DU})$	$O(NA_{DU})$	$O(1)$	$O(1)$	$O(NA_{DU})$	$O(1)$	$O(1)$	$O(NA_{DU})$	$O(NA_T)$	$O(NA_{msa})$	/
FPCA	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(NA_T)$	$O(1)$	$O(1)$

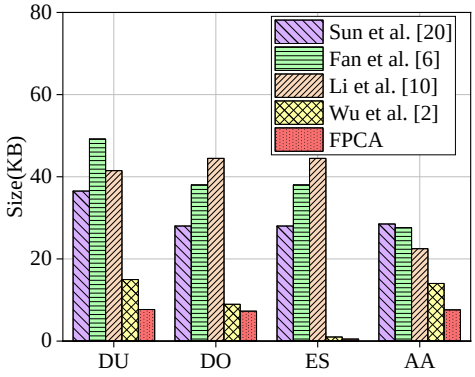


Fig. 4. The Storage Costs

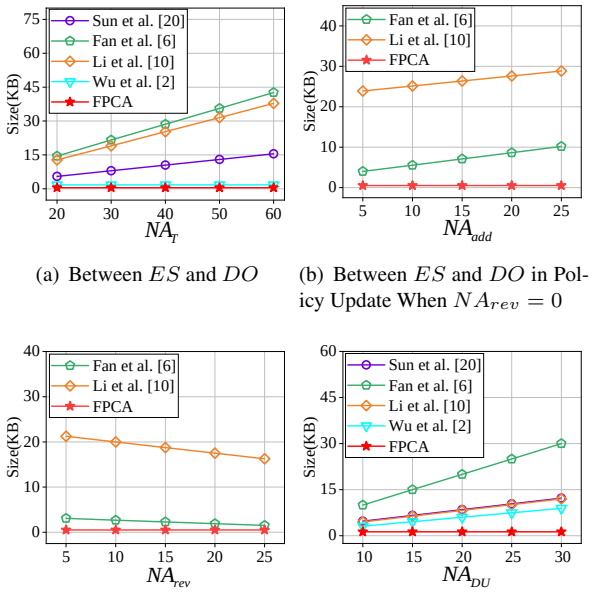
ciphertext and the update message in FPCA is lower than the ciphertext in [2]. Thus, the communication costs of FPCA are lower than other compared schemes.

In Table 5, we present the computation cost. We omit the computation costs for the hash operation and the computation in Z_p, which are negligible. We also omit the symmetric encryption/decryption operation, which has the same running time in all the compared schemes. The multiplication operations in G and G^* have similar processing times, so we use T_G uniformly to represent the multiplication operations of group elements. According to Table 6, the computational complexity of FPCA is $O(1)$ in stages other than encryption, while for other schemes compared it is $O(NA)$. In the encryption phase, the computational complexities of all schemes compared are $O(NA_T)$, only [20] and FPCA does not require bilinear pairing operations, and the computational costs of FPCA are lower than those of [20] by $NA_T \cdot (7T_{EXP} + 4T_G) + 8T_{EXP} + 4T_G$. Therefore, the computational costs of our FPCA are lower than those of other compared schemes.

The cross data domain access for FPCA and other compared schemes primarily involves the transmission and decryption of ciphertext from Dm_2. The complexities of decryption and transmission of ciphertext have been discussed above. In addition, [20] needs to perform conversion calculations on the secret key with computational complexity of $O(NA_{U1})$ before decryption. Thus, the computational complexity of FPCA is lower than other schemes compared for cross data domain access.

6.3. Experimental results

In Fig. 4, we evaluated the storage costs of each entity in FPCA and other compared schemes. We ignore the storage costs of CA and CS because CA only stores the public parameters PP, which we analyze in all other entities, CS stores the encrypted data ED_κ that have the same storage costs in FPCA and other compared schemes. As we can see from Fig. 4, FPCA have lower storage costs than other compared schemes. The storage costs of DU, DO, ES, and AA are only $7.41\ KB$, $7.27\ KB$, $0.5\ KB$, and $7.35\ KB$ in FPCA and are only 1% to 25% of [20], [6], and [10]. Comparing FPCA and [2], the storage costs of FPCA are only 50% to 80% of [2]. This is because the length of PP, the secret key, the ciphertext, and the update message in FPCA are constant.



(a) Between ES and DO

(b) Between ES and DO in Policy Update When $NA_{rev} = 0$

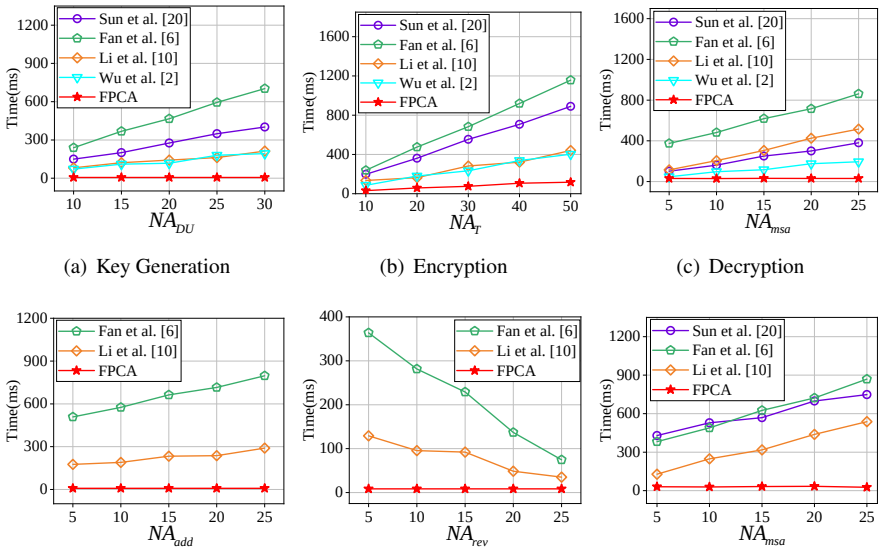
(c) Between ES and DO in Policy Update When $NA_{add} = 0$

(d) Between AA and DU

Fig. 5. The Communication Costs

In Fig. 5, we evaluated the communication costs between the entities in FPCA as well as other compared schemes. We ignore the communication with CA, because it only transmits PP, which we have already analyzed in storage costs, and the length of PP is not significantly different in the FPCA and compared schemes. Figs. 5(a), 5(b) and 5(c) show the communication costs between ES and DO, where FPCA remains at $0.5\,KB$, $0.38\ KB$, and $0.38\ KB$, respectively, and are lower than other comparative schemes. The communication costs between ES and DU are consistent with Fig. 5(a). Fig. 5(d) shows the communication costs between AA and DU, FPCA remains $1.27\ KB$ and is lower than other schemes compared. This is because in FPCA, the lengths of transmitted messages are constant and the length of the ciphertext is lower than [2].

Fig. 6 shows the time costs of various stages of access control in our FPCA and other compared schemes. Except for Figs. 6(a) and 6(b), we keep NA_{DU} and NA_T unchanged. Fig. 6(a) shows the time costs of key generation for FPCA and the compared schemes, FPCA remains around $5.55\ ms$. Fig. 6(b) shows the time costs of encryption for FPCA and the compared schemes, where the time costs of FPCA increase from $31.4\ ms$ to $117.4\ ms$ as NA_T grows. Fig. 6(c) shows the time costs of decryption for FPCA and the compared schemes, FPCA remains around $29.75\ ms$ and is lower than other compared schemes. Figs. 6(d) and 6(e) show the time costs of access policy update for FPCA and the compared schemes, the average running time of FPCA remains around $8.4\ ms$. The time costs of various stages of access control for FPCA are lower than those of other compared schemes. This is because the computational complexities of key generation, decryption, and access policy update for FPCA are $O(1)$, while for other schemes are



(a) Key Generation (b) Encryption (c) Decryption

(d) Access Policy Update When $NA_{rev} = 0$ (e) Access Policy Update When $NA_{add} = 0$ (f) Cross Data Domain Access

Fig. 6. The Time Costs

$O(NA)$. For encryption, although the computational complexity for all the schemes compared is $O(NA_T)$, but only FPCA and [20] have no bilinear pairing operation, and the computational costs of FPCA are lower than [20].

In Fig. 6(f), we evaluated the time costs of accessing data from another data domain. We can see that when NA_{msa} increases, the time costs of FPCA remain around $30.4\ ms$, while for [20], [6], and [10] are increase and higher than FPCA. This is because in FPCA, the computational complexity of decrypting ciphertext from Dm_2 is $O(1)$, and the lengths of the ciphertext and other transmitted messages are constant.

In summary, the experimental results show that the storage, communication, and computation costs of FPCA are lower than those of other compared schemes in all phases of cross data domain access control. Compared with different access control schemes, FPCA has a significant advantage in increasing the efficiency of cross data domain access control and updating the access policy.

6.4. Engineering applications

Cloud-edge collaboration technology finds widespread application in the field of new energy vehicle manufacturing, enabling large-scale data processing and real-time response services for new energy vehicle production. When combined with data access control technology, it allows new energy vehicle manufacturing enterprises to safely share production data. These production data are generated and utilized by numerous IoT devices in various business lines. However, the production of new energy vehicles involves multiple distinct business lines, each belonging to different data domains and possessing unique

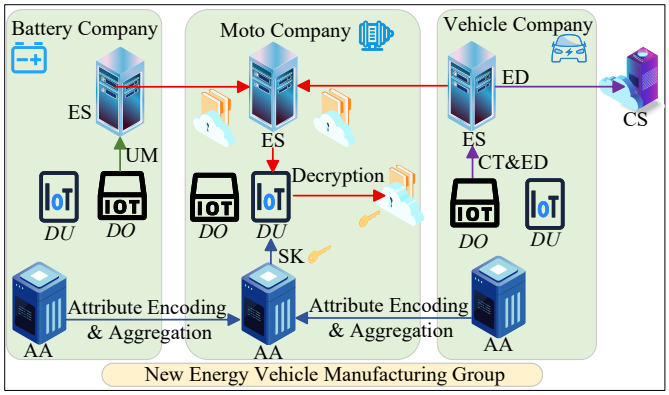


Fig. 7. Application Scenario of FPCA in New Energy Vehicle Manufacturing Industry

access control attribute sets. Among data sharing, DU needs multiple secret keys to access data from different business lines. The large number of attributes can lead to an expansion in secret key and ciphertext lengths, and unable to update data access authority flexibly. These impose a high computational resource overhead on resource-constrained IoT devices in business lines, thereby limiting production efficiency. FPCA enables data access for multi-business lines using one secret key while maintaining constant ciphertext and secret key length, and also reduces the overhead of updating data access authority, achieves dynamic data sharing across business lines, and improves collaborative production efficiency, as shown in Fig. 7.

In Fig. 7, the new energy vehicle manufacturing group comprises a battery company, a motor company, and a vehicle company, each engaged in different businesses. Each company represents a data domain and is equipped with an AA and an ES. Both DU and DO utilize IoT devices on the production line. In Fig. 7, when shared data is generated during the vehicle process, DO of the vehicle company first sets an access policy, then encrypts the vehicle production data to be shared, and uses attribute encoding to produce a constant-length ciphertext CT. DO uploads CT and encrypted data ED to ES of the vehicle company. ES stores CT and uploads ED to the central CS for storage. After registration by DU of the motor company, AA of the motor company generates a constant-length secret key SK containing the attributes of the three companies using attribute encoding and aggregation. DU can use SK to decrypt shared data from various companies, which DU obtains through ES of the motor company. When DO of the battery company wants to adjust the access policy of battery production data, it uses attribute encoding to generate an update message UM of constant-length and uploads it to ES to replace the corresponding part in the ciphertext, thus completing the update.

7. Conclusions and Future Work

In this work, we present FPCA, a fully constant-length and policy-updating cross data domain access control for a cloud-edge collaborative environment. To improve the efficiency of cross data domain access control, we design the MDKG and CLCE algorithms to keep

the length of the secret key and the ciphertext constant and allow the user to access data from other data domains without any conversion operation. We design the APU algorithm to update the access policy with a constant-length update message and low computational complexity to achieve dynamic access control. The security analysis proved the security of our FPCA. The experiment results indicate that our FPCA can achieve efficient cross data domain access control.

However, FPCA cannot hide the access policy, considering that access policies contain user privacy, which may be leaked to adversaries. Moreover, most of the encryption and decryption are performed by IoT devices of IoT users, which are resource-constrained. The low computational efficiency of IoT users will limit the improvement of access control efficiency. Thus, in the future, we will consider adding hidden access policy functionality to FPCA to address the issue of user privacy leakage. Finally, to improve the efficiency of FPCA, we will consider outsourcing most of the encryption and decryption calculations to servers in FPCA.

Acknowledgement. This work is supported by the Scientific Research Fund of Hunan Provincial Education Department (No.24A0337), and the Natural Science Foundation of Hunan Province (No.2025JJ50348 and 2025JJ50399).

References

1. Allison, L., Brent, W.: Unbounded hibe and attribute-based encryption. Lecture Notes in Computer Science 6632, 547–567 (2011)
2. Axin, W., Yinghui, Z., Jianhao, Z., Qiuxia, Z., Yu, Z.: Hierarchal bilateral access control with constant size ciphertexts for mobile cloud computing. IEEE Transactions on Cloud Computing 12(2), 659–670 (2024)
3. Bingcheng, J., Qian, H., Peng, L., Sabita, M., Yan, Z.: Blockchain empowered secure video sharing with access control for vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems 24(9), 9041–905 (2023)
4. Chen, J., Lu, F., Liu, Y., Peng, S., Cai, Z., Mo, F.: Cross trust: A decentralized ma-abe mechanism for cross-border identity authentication. International Journal of Critical Infrastructure Protection 44, 100661 (2024)
5. Chen, L., Chen, Y., Liang, W., Li, X., Li, K.C., Wang, J., Xiong, N.: Mass: A multi-attribute sketch secure data sharing scheme for iot wearable medical devices based on blockchain. IEEE Internet of Things Journal 12(2), 1990–2001 (2025)
6. Fan, H., Li, Q., Xiong, J., Li, R., Chen, W., Huang, H.: Decentralized access control for privacy-preserving cloud-based personal health record with verifiable policy update. IEEE Internet of Things Journal 11(9), 16887 – 16901 (2024)
7. Fan, Y., Liu, S., Tan, G., Lin, X.: Cscac: One constant-size cpabe access control scheme in trusted execution environment. International Journal of Computational Science and Engineering 19(2), 162–168 (2019)
8. Huai, Q., Yuan, W., Wu, Y., Fan, P.: Downlink otfs-rsma cross-domain transmission scheme and sum-rate maximization. IEEE Communications Letter 29(3), 600–604 (2025)
9. Khandla, D., Shahy, H., Bz, M.K., Pais, A.R., Raj, N.: Expressive cp-abe scheme satisfying constant-size keys and ciphertexts. Cryptology ePrint Archive 2019, 1257 (2019)
10. Li, J., Zhang, E., Han, J., Zhang, Y., Shen, J.: Ph-mg-abe: A flexible policy-hidden multi-group attribute-based encryption scheme for secure cloud storage. IEEE Internet of Things Journal 12(2), 2146 – 2157 (2024)

11. Meng, X., Liang, W., Xu, Z., Li, K.C., Khan, M.K., Kui, X.: An anonymous authenticated group key agreement scheme for transfer learning edge services systems. ACM Transactions on Sensor Networks 20(3), 1–23 (2024)
12. Mosteiro-Sanchez, A., Barcelo, M., Astorga, J., Urbieta, A.: End to end secure data exchange in value chains with dynamic policy updates. Future Generation Computer Systems 158, 333–345 (2024)
13. Papatsimouli, M., Lazaridis, L., Ziouzios, D., Dasygenis, M., Fragulis, G.: Internet of things (iot) awareness in greece. SHS Web of Conferences. EDP Sciences 139, 03013 (2022)
14. S., A.S., Han, R., Rudolph, C., Grobler, M.: Dacp: Enforcing a dynamic access control policy in cross-domain environments. Computer Networks 237, 110049 (2023)
15. Shiwen, Z., Jiayi, H., Wei, L., Keqin, L.: Mmds: A secure and verifiable multimedia data search scheme for cloud-assisted edge computing. Future Generation Computer Systems 151, 32–44 (2024)
16. Shiwen, Z., Yibin, Y., Wei, L., Arthur, S.V.K., Guoqi, X., Kim-Kwang, R.: Mkss: An effective multi-authority keyword search scheme for edge-cloud collaboration. Journal of Systems Architecture 144, 102998 (2023)
17. Shiwen, Z., Ziwei, Y., Wei, L., Kuan-Ching, L., Ciprian, D.: Baka: Biometric authentication and key agreement scheme based on fuzzy extractor for wireless body area networks. IEEE Internet of Things Journal 11(3), 5118–5128 (2023)
18. Shiwen, Z., Ziwei, Y., Wei, L., Kuan-Ching, L., Di Martino, B.: Bcae: A blockchain-based cross domain authentication scheme for edge computing. IEEE Internet of Things Journal 11(13), 24035–24048 (2024)
19. Su, X., An, L., Cheng, Z., Weng, Y.: Cloud–edge collaboration-based bi-level optimal scheduling for intelligent healthcare systems. Future Generation Computer Systems 141, 28–39 (2023)
20. Sun, J., Xu, G., Li, H., Zhang, T., Wu, C., Yang, X.: Sanitizable cross-domain access control with policy-driven dynamic authorization. IEEE Transactions on Dependable and Secure Computing pp. 1–17 (2025)
21. Tianqiao, Z., Mingming, J., Fucai, L., Yuyan, G.: A lattice-based puncturable cp-abe scheme with forward security for cloud-assisted iot. IEEE Internet of Things Journal 12(14), 26538–26554 (2025)
22. Vipul, G., Omkant, P., Amit, S., Brent, W.: Attribute-based encryption for fine-grained access control of encrypted data. Proceedings of The 13th ACM Conference on Computer and Communications Security pp. 89–98 (2006)
23. Wang, P.B., Li, K.C., Shi, R.H., Shao, B.S.: Vc-dcps: Verifiable cross-domain data collection and privacy-persevering sharing scheme based on lattice in blockchain-enhanced smart grids. IEEE Internet of Things Journal 10(14), 12449–12461 (2023)
24. Xue, J., Shi, L., Zhang, W., Li, W., Zhang, X., Zhou, Y.: Poly-abe: A traceable and revocable fully hidden policy cp-abe scheme for integrated demand response in multi-energy systems. Journal of Systems Architecture 143, 102982 (2023)
25. Yang, M., Wang, H., Wan, Z.: Pul-abe: An efficient and quantum-resistant cp-abe with policy update in cloud storage. IEEE Transactions on Services Computing 17(3), 1126–1139 (2024)
26. Yang, R., He, H., Xu, Y., Xin, B., Wang, Y., Qu, Y., Zhang, W.: Efficient intrusion detection toward iot networks using cloud–edge collaboration. Computer Networks 228, 109724 (2023)
27. Yannis, R., Brent, W.: Practical constructions and new proof methods for large universe attribute-based encryption. Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security pp. 463–474 (2013)
28. Ying, Z., Jiang, W., Liu, X., Xu, S., Deng, R.H.: Reliable policy updating under efficient policy hidden fine-grained access control framework for cloud data sharing. IEEE Transactions on Services Computing 15(6), 3485–3498 (2021)
29. Zhang, S., Ren, F., Liang, W., Li, K., Ling, N.: Gpvo-fl: Grouped privacy-preserving and verification-outsourced federated learning in cloud-edge collaborative environment. IEEE Transactions on Network and Service Management 22(5), 4175 – 4191 (2025)

30. Zuo, Y., Xu, L., Li, J., Li, J., Wang, X., Piran, M.J.: Secure and efficient blockchain-based access control scheme with attribute update. IEEE Transactions on Consumer Electronics 71(1), 1539 – 1550 (2024)

Shiwen Zhang received his B.S. degree in Information and Computing Science from the University of Changsha in 2010, and received his Ph.D. degree from the College of Computer Science and Electronic Engineering, Hunan University, in 2016. He is an associate professor at the School of Computer Science and Engineering, Hunan University of Science and Technology. He is a senior member of IEEE and CCF. His research interests include security and privacy issues in cloud computing, privacy protection, and information security.

Siwei Wen received a B.S. degree from Xiangtan University and is currently pursuing an M.S. degree from the School of Computer Science and Engineering at Hunan University of Science and Technology. His research interests include access control and privacy protection in cloud and edge computing.

Wei Liang received a Ph.D. degree in computer science and technology from Hunan University in 2013. He was a Post-Doctoral Scholar at Lehigh University from 2014 to 2016. He is currently a Professor and the Dean of the School of Computer Science and Engineering at Hunan University of Science and Technology, China. He has authored or co-authored more than 140 journal/conference papers. His research interests include network security, cloud computing, and security management in edge computing.

Received: February 08, 2026; Accepted: February 22, 2026.