Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

A group key exchange and secure data sharing based on privacy protection for federated learning in edge-cloud collaborative computing environment

Wenjun Song[1] | Mengqi Liu[1] | Thar Baker[2] | Qikun Zhang[1] | Yu-an Tan*[3]

[1]School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, China
[2]School of Architecture, Technology and Engineering, The University of Brighton, Brighton BN2 4GJ, UK
[3]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China

Correspondence
Yu-an Tan, School of Computer Science and Technology, Beijing Institute of Technology, No. 5 Zhongguancun South Street, Haidian District, Beijing, China. Email: tan2008@bit.edu.cn

Summary

Federated Learning (FL) is widely used in IoT scenarios such as health research, automotive autopilot, and smart home systems. In the process of model training of FL, each round of model training requires rigorous decryption training and encryption uploading steps. The efficiency of FL is seriously affected by frequent encryption and decryption operations. A scheme of key computation and key management with high efficiency is urgently needed. Therefore, we propose a group key agreement technique to keep private information and confidential data from being leaked, which is used to encrypt and decrypt the transmitted data among IoT terminals. The key agreement scheme includes hidden attribute authentication, multi-policy access, and ciphertext storage. Key agreement is designed with edge-cloud collaborative network architecture. Firstly, the terminal generates its own public and private keys through the key algorithm; then confirms the authenticity and mapping relationship of its private and public keys to the cloud server. Secondly, IoT terminals can confirm their cryptographic attributes to the cloud and obtain the permissions corresponding to each attribute by encrypting the attributes. The terminal uses these permissions to encrypt the FL model parameters and uploads the secret parameters to the edge server. Through the storage of the edge server, these ciphertext decryption parameters are shared with the other terminal models of FL. Finally, other terminal models are trained by downloading and decrypting the shared model parameters for the purpose of FL. The performance analysis shows that this model has a better performance in computational complexity and computational time compared with the cited literature.

KEYWORDS:
IoT, Edge-cloud collaborative, Federated learning, Group key agreement, Privacy protection

1 | INTRODUCTION

With the development of the industrial level, the original industrial service capacity is more and more limited[1], and the current stage of industrial development has put forward higher requirements for the implementation period[2]. Therefore, edge-cloud

collaborative network is widely used in smart transportation[3], smart agriculture, manufacturing of industrial IoT, etc. The application of these scenarios requires that the terminals in the system work together to accomplish specific tasks through collaborative computing, collaborative operations, and resource sharing[4]. In collaborative computing, the group session key is generated by the requester. The requester uses this group session key to encrypt the data into ciphertext and then broadcasts the secret message to the group. Legitimate service providers can calculate the group session key and decrypt the ciphertext data. After the data is processed by the service provider, the data is again encrypted by the group session key. The processed ciphertext data is returned to the requester. Terminals are concerned about the leakage of secret data in the process of data sharing and collaborative computing. However, the performance and efficiency of the whole system collaborative computing will be affected if the data is not shared with other terminals. And Federated Learning (FL) is an emerging technology underlying artificial intelligence. The goal is to carry out efficient machine learning among multiple participants or multiple computing nodes to guarantee information security when exchanging big data, protect terminal data and personal data privacy, and ensure compliance with legal regulations[5]. Due to the excellent characteristics of FL for the confidentiality of shared data and the efficiency of execution, some intelligent application areas of machine learning can be combined with it, giving the conditions for intelligent computing and intelligent operation in edge cloud collaborative application scenarios.

Secure data sharing and privacy protection issues are involved in the process of federated machine learning[6]. For example, by using encryption-based user sample alignment techniques, the system unites users' characteristics for modeling without exposing the shared users of the two models that do not disclose their respective data and without exposing users that do not overlap with each other. After the participants have trained the original model, the local model parameters will be encrypted and transmitted to the rest of data holders participating in the joint training[7]. Therefore, assuming that there are n participants in this joint training, each participant needs to transmit the encryption model parameters at least $2(n-1)$ times[8]. Considering the issues of identity confirmation and privacy protection of identity information, key computation and key negotiation for model parameter encryption, group key agreement and secure data sharing techniques are proposed. The main contributions of the paper are as follows.

1) Self-confirmation of key generation: Securing the private key is crucial because it is an important basis for determining the identity of the terminal. Traditional key generation algorithm mainly relies on a trusted third party to generate a private key based on the user's identity information and then distribute it to the user. There are many potential risks in this scheme. For example, the adversary attacks the third party to get the user's private key and then impersonates the terminal to do illegal operations. In addition, the terminal private key is vulnerable to eavesdropping and other security threats during the transmission process. Aiming at these problems, we propose the key self-verification algorithm, i.e., each terminal in IoT calculates its own public/private key pair by the key generation algorithm itself, and then confirms to the cloud server that the key pair is its own. The key self-verification algorithm avoids the security risks of traditional schemes because the key is generated by terminal itself, and the algorithm is used to verify the correspondence between its identity and the key.

2) Authentication with hidden attributes: Identity-based authentication tends to leak the identity information of terminal, but attribute-based authentication can infer identity information through attributes. Therefore, we propose the hidden attribute authentication method, where the terminal of IoT firstly encrypts the attributes and then sends the ciphertext attributes to the cloud server for ciphertext attribute authentication. Hidden attribute authentication is used to avoid the leakage of identity and attribute information of the terminal during transmission.

3) Ciphertext model parameters sharing: The training model of each IoT terminal encrypts the trained model parameters of its own, and then stores the ciphertext parameters on the edge server. Other training models can download and decrypt the shared model training parameters from the edge server, and use them for federated learning to achieve optimal model training. The security and anti-leakage of shared model data are guaranteed.

4) Self-adaptive model data sharing: In edge-cloud collaborative application scenarios, each terminal may participate in several different model training tasks, and the parameters of each training model are only limited to be shared in the federation of a certain task. Therefore, the parameter sharing for multiple training models requires different data sharing permissions. For different attribute constraint algorithms of data, the terminal self-adaptively combines various access rights according to its own attribute rights. The data is shared securely by self-adapting to the data permissions.

The other sections of the manuscript are ordered in the following way. Section 2 discusses the Related Work, followed by Section 3, we introduce the basics used in this article; We describe the system architecture model in Section 4 and introduce the detailed process of the model in Section 5; In sections 6 and 7 we respectively analyze the security and performance of the proposed model; Finally, we conclude the paper in Section 8.

2 | RELATED WORK

Wireless network data transmission is adopted by most application scenarios of smart internet of things (IoT). This method is not only vulnerable to cyber attacks, data can be easily intercepted or leaked, and mobile terminals are easily controlled by adversaries. The security of data sharing and terminal privacy protection is a challenging research project in this application scenario. In recent years, many scholars have proposed solutions for privacy leakage during information transfer in different application scenarios.To achieve decentralized file access, Lin et al.[9] added a data access mechanism in Interplanetary File System (IPFS). At the same time, blockchain is used to store file information and user permissions. File data sharing can be managed by the improved IPFS according to user permissions. Privacy protection is achieved.Bin et al.[10] proposed distributed K-means clustering with differential privacy and homomorphic encryption by combining blockchain technology. The scheme is designed to address the security vulnerability in federal learning, which makes private data more secure.Secure multi-party computing can achieve privacy protection in multi-party data sharing. However, traditional secure multi-party computing is inefficient and the computation protocol is complex. Yang et al.[11] propose a blockchain-based secure multi-party computation architecture, which has excellent properties of decentralization, verifiability and high reliability in the privacy protection process. Rao et al.[12] proposed a privacy-preserving multi-group data sharing scheme in the cloud with group signatures and broadcast encryption in the case of multiple groups of shared data. Group signatures and broadcast encryption are used to efficiently revoke users during anonymous intra-group and cross-group data sharing. Wang et al.[13] propose a verifiable thresholded multi-secret sharing privacy preservation scheme for scenarios without secure channels. Multiple secrets can be shared among a group of participants and it is possible to detect if a trader or participant is spoofing. Also works in environments where secure channels are not available. Piao et al.[14] designed a model to address the characteristics of diverse types and complex attributes of shared data in government departments. First clustering algorithm is used to reduce the dimensionality of the data. Then k-medoids clustering algorithm is used to improve data availability. Finally, the clustering results are anonymized using generalization techniques to ensure data privacy. In addition, Piao et al.[15] proposed an improved Local Difference Privacy (LDP) based scheme. Data chunking technique and count mean sketch (CMS) algorithm were adopted in this scheme. It overcomes the disadvantage of requiring strict data size in case of large data fields.

New schemes are proposed in the literature [16-24] for the authentication part of the data sharing process. Xuan et al.[16] propose an optimized scheme for cross-domain authentication between heterogeneous IoT applications using certificate-less public key cryptography and smart contract technology. The scheme has better performance in terms of communication volume and verification computation cost. The capability of cross-domain authentication is improved. Braeken et al.[17] proposed a scheme to implement terminal self-certified. In the scheme, users share the same private public key pair and use the Canetti-Krawczyk (CK) secure mutual authentication protocol. To prevent data leakage and data propagation delay, Zhao et al.[18] design a federated learning collaborative authentication protocol for shared data with low authentication delay. Efficient anonymous mutual authentication and key negotiation are implemented. Due to the immutability and traceability of blockchain technology, many literatures use this technique in identity authentication. Fan et al.[19] proposed a secure and efficient authentication and data sharing scheme for IoT based on blockchain technology. While ensuring efficient authentication, the authenticity of participants' identities can be ensured. Chaitanya et al.[20] also used blockchain technology to achieve bidirectional authentication with bilinear mapping knowledge in the authentication phase. A scheme is given for the danger of third-party trust centers, which also reduces the average communication time and cost. Jia et al.[21] propose a scheme that uses edge computing to decentralize authentication requests. By verifying the identity of IoT devices through blockchain, the overhead caused by the authentication phase is greatly reduced. Lv et al.[22] proposed a blockchain-based cross-domain authentication scheme. In this scheme, the identity generation, cryptographic storage, registration, joining and exiting of terminals are controlled by themselves. The authenticity and trustworthiness of these identities are ensured by establishing a trusted identity checking mechanism with minimal computational cost. Through blockchain smart contracts and consensus mechanism, Zhan et al.[23] proposed a blockchain-based distributed CA identity authentication between subjects. Mutual authentication of transnational identities can be achieved, which is of great significance for transnational data sharing and privacy security. Wu et al.[24] combined blockchain with biometrics to form a shared session key during telemedicine to protect the patient's privacy. The scheme is highly secure and practical.

In order to achieve fine-grained data access control, attribute-based encryption has become a mainstream encryption technique favored by many scholars. Rohit et al.[25] extended the ciphertext policy attribute based encryption to achieve scalability and fine-grained access control. The protocol has good scalability and high security. Miguel et al.[26] proposed a scheme that can solve secure sharing of cloud storage data. Based on attribute-based encryption(ABE), the encrypted data is stored, retrieved and shared in the cloud. The access control mechanism of encrypted data and information retrieval task is triggered by search

access control to secure the data during the sharing process. Nancy et al.[27] combined blockchain technology and proposed a scheme using encryption based on multi-authority attributes to improve efficiency of data sharing. User attributes are hidden by pre-encryption technique to ensure data security when it is shared. Arasi et al.[28] proposed an auditable attribute-based data access control scheme. The scheme combines blockchain and attribute-based access control with traceability. Data integrity is guaranteed and efficient data sharing is achieved. To address the problems of high computational overhead and dynamic management, Ge et al.[29] proposed a decentralized attribute-based data sharing scheme. The scheme hides identity information and supports user revocation and authentication. Secure sharing and data confidentiality is guaranteed. The scheme proposed by Ye et al.[30] uses searchable attribute-based cryptography. Resource keywords can be updated in the sharing phase and no interaction with the key generation center is required. It has better resistance to chosen ciphertext attack and chosen keyword attacks.

Recent researches have made great contributions to the security technology of data sharing, but further improvements and advances are needed for secure data sharing in new information technologies and their application scenarios. In this paper, we propose a multi-policy secure ciphertext data sharing model based on hidden attribute authentication. Private key leakage caused by an illegal party intercepted is avoided during third-party key distribution. Encrypted attribute authentication technology is not only used to achieve identity authentication, but also effectively protect the terminal's identity and attribute information from being leaked. Secure and flexible multiple access policies are used for data access control to ensure the practicality of data sharing.

3 | PRELIMINARIES

3.1 | Bilinear mapping

First, the definition of bilinear mapping is given. Let G_1 be an additive group and its generator element be g_1. Let G_2 be a multiplicative cyclic group, G_1 and G_2 have the same large prime order q, where $q \geq 2^k + 1$ (k is the security parameter). Computing discrete logarithms on G_1 and G_2 is difficult. Groups G_1 and G_2 are a pair of bilinear groups, and e is a computable bilinear mapping function, i.e., $e : G_1 \times G_1 \rightarrow G_2$. e satisfies the follow properties:

(1) Bilinearty: for any $g_1, g_2 \in G_1$ and $a, b \in \mathbb{Z}_q^*$, there is $e(ag_1, bg_2) = e(g_1, g_2)^{ab}$.

(2) Non-degeneracy: that is $e(g_1, g_2) \neq 1$.

(3) Computability: there exists a valid algorithm to efficiently compute the value of $e(g_1, g_2)$ for any $g_1, g_2 \in G_1$.

3.2 | Computational hardness assumption

The following assumptions exist in the additive group G_1:

(1) Discrete Logarithm Problem (DLP). Suppose $g_1, g_1' \in G_1$, that solving for an unknown $a \in \mathbb{Z}_q^*$ makes $g_1' = ag_1$ computationally difficult.

(2) Computational Diffie-Hellman Problem (CDHP). The generator g_1 of group G_1 and elements (ag_1, bg_1) are known, where $a, b \in \mathbb{Z}_q^*$, and it is difficult to compute abg_1 when a and b are unknown.

(3) Computing Bilinear Diffie-Hellman Problem (CBDHP). Given elements $g_1, g_1^a, g_1^b, g_1^c \in G_1$, where $a, b, c \in \mathbb{Z}_q^*$, it is difficult to calculate $e(g_1, g_1)^{abc}$.

(4) Inverse Computing Diffie-Hellman Problem (Inv-CDHP). For any $ag_1, abg_1 \in G_1$, where $a, b \in \mathbb{Z}_q^*$. Knowing g_1, ag_1, abg_1, it is difficult to compute $(ab/a)g_1$.

Table 1 illustrates symbols that appear below.

3.3 | Access structure of the model

The specific representation of access policy is access structure, and access tree is a very flexible access structure[31]. Access structure of the model is a multinomial tree that defines logical structure of data access rights, which consists of system-defined attributes and logical gates.

Access Structure: Let set $D = \{d_1, d_2, \cdots, d_n\}$ denote the attribute domain, and access structure be denoted by $\mathbb{S} \subseteq 2^{\{d_1, d_2, \cdots, d_n\}} \neq \varnothing$. If there exist sets B and C, and $B \in \mathbb{S}$ can be derived from $C \in B$ and $C \in \mathbb{S}$, then the set $\mathbb{S}$ is said to be monotone. Attribute set that conforms to the access policy is called the authorized set, and attribute set that does not conform to the access policy is called the unauthorized set.

Table 1 Symbol Description.

Symbol	Description
G_1	Additive groups on elliptic curves
G_2	Multiplicative groups on elliptic curves
g_1	Generator of G_1
e	Computable bilinear mapping
$H_1(\cdot), H_2(\cdot), H_3(\cdot)$	Collision-resistant hash functions
$\|$	Connection symbol
$\oplus$	Exclusive OR
PK_A, SK_A	Public and private key of Cloud
pk, sk	Public and private key of terminal

Access Tree: In any non-empty access tree, the structure of access policy is represented by the tree V, x is the node of tree V, and the subtree with x as the root node is represented by V_x. If x is a non-leaf node, then this node represents a threshold. num_x denotes the number of child nodes of the subtree V_x, k_x is the threshold value of node x, where $k_x \in [1, num_x]$.

(1) When $k_x = 1$, node x is represented as an OR gate;

(2) When $k_x = num_x$, node x is represented as an AND gate;

(3) When $k_x < num_x$, node x is represented as an OF gate and the value of k_x is the number of attribute nodes to be satisfied. For example, $n\ of\ (\sim)$ means that at least n attributes need to be owned in $(\sim)$.

4 | SYSTEM STRUCTURE MODEL

A multi-policy data sharing model based on hidden attribute authentication is proposed, which contains three entities: Cloud, Edge Server (ES) and network terminal (or user). The model is proposed that terminals participating in sharing need to authenticate the attribute information to the Cloud and obtain the authority corresponding to the attribute after authentication. After the Cloud distributes attribute authority to the terminals, it records relevant information in the table of terminal authentication information and shares the table with the Edge Server. Data sharer stores encrypted resources in the Edge Server, and the server updates the corresponding information in the secure data sharing platform. Data viewer verifies the permission with the Edge Server, and the Edge Server verifies the viewer's identity according to the secure data sharing platform. After verification by the Edge Server, viewer can obtain the ciphertext resource, as shown in Figure 1. The components and process of the model are described below:

1) The Cloud is a reliable and trusted center that is responsible for generating system public parameters and master keys; then authenticating, registering and distributing attribute keys to terminals; and maintaining the table of user authentication information, which contains identity information, permission information and public key of the successfully authenticated terminal.

2) Edge Server has huge storage space and powerful computing power, which are mainly used for storing shared resources. Information about the resource is registered in the secure data sharing platform. Verify the identity of the resource visitor by accessing the sharing platform, and return the ciphertext resource to the visitor with successful authentication.

3) Terminals are the participants of data sharing, they can be both the sharers and the demanders of resources. Only users who have been authenticated by the Cloud and have attribute rights can upload or download resources.

4) The secure data sharing platform records information about shared resources, including the identity of the uploader, public key, hash value of the shared cryptographic data, cryptographic keywords, cryptographic function coefficients, and access policy of the resources. Many Edge Servers share one secure data sharing platform.

Step ①-②: After the initialization of the system, all terminals need to register with the Cloud for authentication. After the Cloud legitimatizes the private key of terminal, it sends corresponding attribute authority secret to the terminal according to the attributes owned by the terminal. Terminals that join later also need to complete this process. Step ③-④: The Cloud keeps the identity information of all terminals and records it in the terminal authentication information table, which includes the valid status, public key and attribute weights of terminals. The information table is shared with the security data sharing platform, but the platform only has permission to view and cannot modify contents of the table. Step ⑤: Terminal that wants to

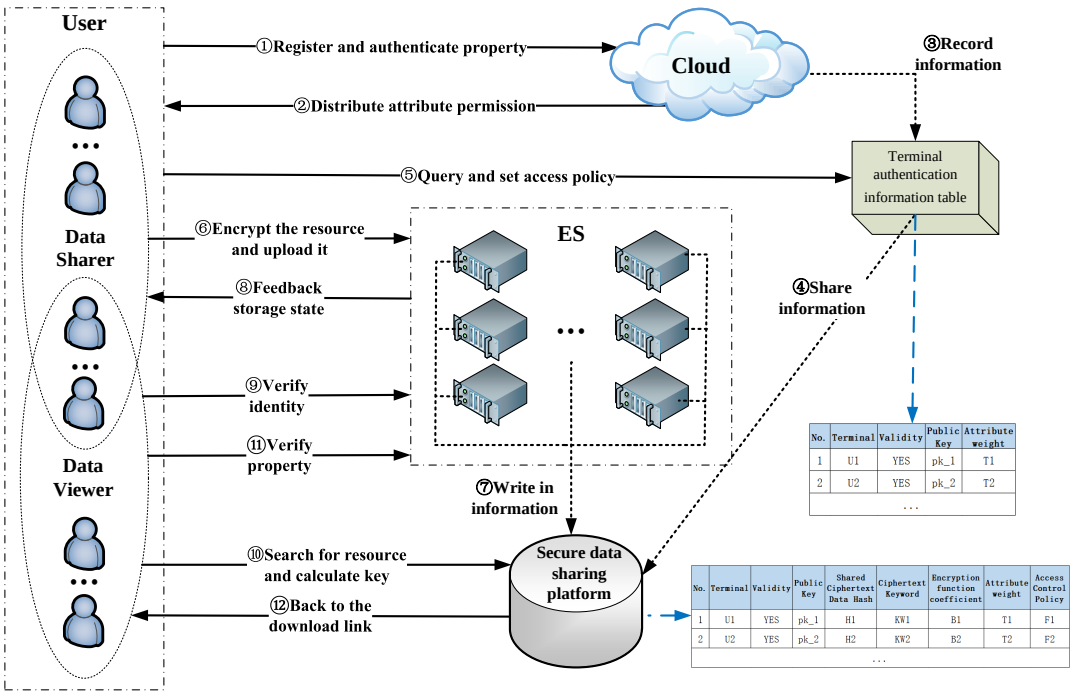

Figure 1 System Model.

share the resource needs to identify the shared object by querying the terminal authentication information table. By analyzing permissions of shared objects, the corresponding access policy is set. The access policy contains multiple attribute weight combinations. Each attribute weight has a corresponding attribute key combination value. Step ⑥-⑧: The sharer randomly selects a encryption key and constructs a polynomial using the combined values of attribute keys obtained in Step ⑤. The resources and resource keywords are encrypted and sent to the Edge Server. The cloud server receives the information sent by the terminal and examines the information. If the authentication passes, the information is written to the secure data sharing platform, and "Upload successful" is returned to the terminal. If the verification fails, step ⑦ will not be executed and "Upload failed" will be returned to the terminal. Step ⑨-⑩: To download the resources, the requestor first needs to prove legal identity to the Edge Server. The terminal searches the secure data sharing platform for a ciphertext with matching privileges, and then calculates the decryption key using the attribute weights and function coefficients. Decrypt and correlate the ciphertext keywords with this key. (The combination of attribute weights varies from one viewer to another, and the computed key combination is not the same, but the same decryption key can be derived.) Step ⑪-⑫: If the keyword is highly relevant to the requested resource, the data requestor needs to further verify the permission with the Edge Server to download. ES checks attribute rights of the requestor. If the permission level is equal to or higher than the level of the ciphertext resource, the download link will be sent to the requesting terminal. At this point, the whole process of data sharing is completed.

5 | SECURE RESOURCE SHARING ACCESS CONTROL

5.1 | Initialization

Assume that there are n terminals in the protocol, denoted by $U = \{u_1, u_2, ..., u_n\}$, and the corresponding set of n terminals' identities is denoted by $ID = \{id_{u_1}, id_{u_2}, ..., id_{u_n}\}$. The Cloud defines an ordered set of network attributes $Attr = \{A_1, A_2, ..., A_j, ..., A_R\}$ and orders them according to their weights, where $A_j < A_{j+1} (j < R, j \in N^*, R \in N^*)$. R is the number of network attributes. In addition, $attr_i = \{a_{i,1}, a_{i,2}, ..., a_{i,r}\}, (1 \le r \le R)$ is the ordered set of attributes of the terminals u_i in the network, where $a_{i,r-1} < a_{i,r}, 1 \le r \le R, r \in N^*, attr_i \subseteq Attr$. r is the rth attribute of the terminal u_i.

Suppose G_1 is an additive group and G_2 is a multiplicative group, and they have the same large prime order q. The discrete logarithms on the additive group G_1 and the multiplicative cyclic group G_2 are difficult. $g_1 \in G_1$ is the generating element of G_1. $e : G_1 \times G_1 \to G_2$ is a computable bilinear mapping. $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*, H_2 : G_1 \to \mathbb{Z}_q^*$ and $H_3 : G_2 \to \{0,1\}^L$ are three hash functions. The system parameters are $params = (q, G_1, G_2, g_1, e, H_1, H_2, H_3)$.

5.2 | Key self-confirmation algorithm

Let $KeyGen(1^{\lambda}) \rightarrow (sk, pk)$ be the key generation function, where λ is the input security parameter and (sk, pk) is the output public/private key pair. Terminal $u_i (1 \leq i \leq n)$ needs to register its identity before entering the system. Assume that the identity information $id_{u_i} (1 \leq i \leq n)$ of all terminals is known by the Cloud. The steps are as follows:

1) The Cloud runs the key generation algorithm $KeyGen(1^{\lambda})$ to obtain the public/private key pair (SK_A, PK_A), where $SK_A \in \mathbb{Z}_q^*$, $PK_A = SK_A g_1$.

2) Terminal $u_i \in U (1 \leq i \leq n)$ chooses a random positive integer $s_{u_i} \in \mathbb{Z}_q^*$, and calculates $sk_{u_i} = H_1(id_{u_i}) s_{u_i}$, $PK_{u_i} = s_{u_i} PK_A$. sk_{u_i} is terminal u_i' private key. $pk_{u_i} = g_1 sk_{u_i}$ is u_i' public key. u_i sends message $\{PK_{u_i}, pk_{u_i}\}$ to the Cloud for authentication of identity and public key.

3) After the Cloud receives message $\{PK_{u_i}, pk_{u_i}\}$ from u_i, it calculates $\mathrm{H}_{u_i} = H_1(id_{u_i}) g_1$ and $\mathrm{Term}_{u_i} = SK_A^{-1} PK_{u_i} = s_{u_i} g_1$ according to the identity of u_i, and verifies whether the equation $e(\mathrm{Term}_{u_i}, \mathrm{H}_{u_i}) = e(pk_{u_i}, g_1)$ is true.

4) The Cloud publishes the public key pk_{u_i} of u_i and completes the key authentication. So far, u_i has the legitimate public/private key pair (sk_{u_i}, pk_{u_i}), as shown in Figure 2.

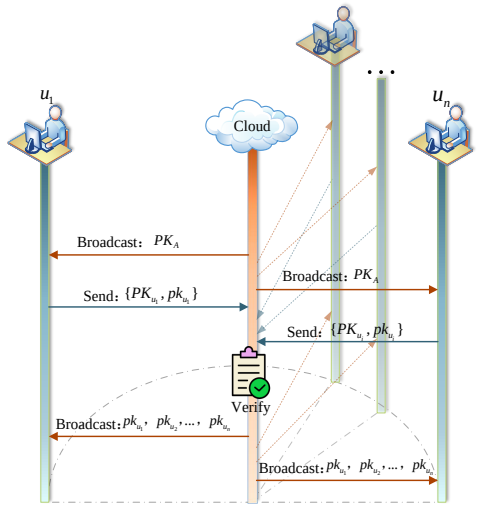

Figure 2 Terminal key verification process.

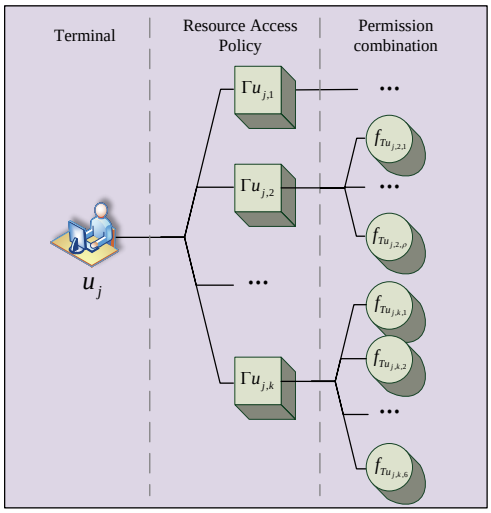

Figure 3 Resource access privilege structure.

The Cloud obtains the identity and public key information of all terminals, and constructs an terminal authentication information table (as shown in Table 2) to record this information. The attribute weight information is added in the subsequent steps.

Table 2 Authentication information of the terminal.

Terminal	u_1	u_2	...	u_n
Validity	yes	yes	...	yes
Public Key	pk_{u_1}	pk_{u_2}	...	pk_{u_n}
Attribute weight	$\{T_{1,1}, T_{1,2}, T_{1,3}\}$	$\{T_{2,1}, T_{2,2}, ..., T_{2,9}\}$	...	$\{T_{n,1}, T_{n,2}, ..., T_{n,r}\}$

5.3 | Ciphertext attribute authentication and attribute permission acquisition

Suppose the sequence number of the ordered attribute set $Attr = \{A_1, A_2, ..., A_j, ..., A_R\}$ is $Ser = \{S_1, S_2, ..., S_j, ..., S_R\}$, and different attribute weight parameters $t_1, t_2, ..., t_R \in \mathbb{Z}_q^*$ are randomly selected for each attribute. u_i sends attributes $\{a_{i,5}, a_{i,3}, ..., a_{i,r}\}$ to the Cloud secretly, and the Cloud returns the corresponding sequence number $ser = \{S_5, S_3, ..., S_r\}$ to the terminal. After receiving the sequence number, u_i compares it with the attributes it owns, and sorts the set of attributes according to that sequence. The sequence of attributes after ranking is $attr_i = \{a_{i,1}, a_{i,2}, ..., a_{i,r}\}$. All terminals are required to complete the process. Then:

1) Suppose the ordered attribute set of u_i is $attr_i = \{a_{i,1}, a_{i,2}, ..., a_{i,r}\}(1 \le r \le R)$. u_i chooses a random number and compute $(\lambda_i g_1, a_{i,1}\lambda_i g_1, a_{i,2}\lambda_i g_1, ..., a_{i,r}\lambda_i g_1)$, $\mathrm{as}_{u_i} = H_2(a_{i,1}\lambda_i g_1||a_{i,2}\lambda_i g_1||...||a_{i,r}\lambda_i g_1)$ and $\beta_i = (a_{i,1} + a_{i,2} + ... + a_{i,r})sk_{u_i}^{-1}\lambda_i g_1$ according to the set of attributes it has. Then u_i sends the message $\{(\lambda_i g_1, a_{i,1}\lambda_i g_1, a_{i,2}\lambda_i g_1, ..., a_{i,r}\lambda_i g_1), \mathrm{as}_{u_i}, \beta_i, pk_{u_i}\}$ to the Cloud;

2) The Cloud calculates $\gamma_i = a_{i,1}\lambda_i g_1 + a_{i,2}\lambda_i g_1 + ... + a_{i,r}\lambda_i g_1$ and $\mathrm{as}'_{u_i} = H_2(a_{i,1}\lambda_i g_1||a_{i,2}\lambda_i g_1||...||a_{i,r}\lambda_i g_1)$ after receiving the message $\{(\lambda_i g_1, a_{i,1}\lambda_i g_1, a_{i,2}\lambda_i g_1, ..., a_{i,r}\lambda_i g_1), \mathrm{as}_{u_i}, \beta_i, pk_{u_i}\}$ from terminal u_i, and then verifies whether the following two equations hold:

$$\mathrm{as}'_{u_i} = \mathrm{as}_{u_i} \tag{1}$$

$$e(\beta_i, pk_{u_i}) = e(\gamma_i, g_1) \tag{2}$$

The purpose of verifying the equality (1) is to prevent man-in-the-middle attack and the parameters from being tampered with during transmission. The purpose of verifying the equality (2) is to verify the identity of terminal through signatures.

3) If equations (1) and (2) are verified, the Cloud calculates equation (3) based on the set of system attributes and verifies equation (4). If equation (4) holds, show that $attr_i \subseteq Attr$. (In equation (4), *or* means 'or'. For example: $(\cdot) = (Val_{A_1}\ or\ Val_{A_2})$ means $(\cdot) = Val_{A_1}$ or $(\cdot) = Val_{A_2}$.)

$$\begin{cases} H_2(A_1\lambda_i g_1) \oplus H_2(A_2\lambda_i g_1) \oplus H_2(A_3\lambda_i g_1) \oplus ... \oplus H_2(A_R\lambda_i g_1) = Val_A \\ H_2(A_2\lambda_i g_1) \oplus H_2(A_3\lambda_i g_1) \oplus ... \oplus H_2(A_R\lambda_i g_1) = Val_{A_1} \\ H_2(A_1\lambda_i g_1) \oplus H_2(A_3\lambda_i g_1) \oplus ... \oplus H_2(A_R\lambda_i g_1) = Val_{A_2} \\ \qquad\qquad ... \\ H_2(A_1\lambda_i g_1) \oplus H_2(A_2\lambda_i g_1) \oplus ... \oplus H_2(A_{R-1}\lambda_i g_1) = Val_{A_R} \end{cases} \tag{3}$$

$$\begin{cases} H_2(a_{i,1}\lambda_i g_1) \oplus Val_A = (Val_{A_1}\ or\ Val_{A_2}\ or\ ...\ or\ Val_{A_R}) \\ H_2(a_{i,2}\lambda_i g_1) \oplus Val_A = (Val_{A_1}\ or\ Val_{A_2}\ or\ ...\ or\ Val_{A_R}) \\ \qquad\qquad ... \\ H_2(a_{i,r}\lambda_i g_1) \oplus Val_A = (Val_{A_1}\ or\ Val_{A_2}\ or\ ...\ or\ Val_{A_R}) \end{cases} \tag{4}$$

4) Suppose $H_2(a_{i,l}\lambda_i g_1) \oplus Val_A = Val_{A_j}(1 \le l \le r, 1 \le j \le R)$, the Cloud selects the corresponding weight parameter from the attribute weight parameter $t_1, t_2, ..., t_R \in \mathbb{Z}_q^*$ according to u_i' attribute set $\{a_{i,1}, a_{i,2}, ..., a_{i,r}\}$, denoted as $(t_{i,1}, t_{i,2}, ..., t_{i,r})$. The Cloud calculates the attribute weights $\{T_{i,0} = \lambda_i g_1, T_{i,1} = t_{i,1}T_{i,0}, T_{i,2} = t_{i,2}T_{i,0}, ..., T_{i,r} = t_{i,r}T_{i,0}\}$ of u_i and classifies the authority levels according to the number of attributes, calculates the hash value $\mathrm{H}_{T_i} = H_2(T_{i,1}||T_{i,2}||...||T_{i,r})$ of the attribute weights and the signature $\eta_{i,s} = SK_A^{-1}(t_{i,1} + t_{i,2} + ... + t_{i,r})g_1$ of attribute weights parameter. Then the Cloud sends message $\{(T_{i,1}, T_{i,2}, ..., T_{i,r}), \mathrm{H}_{T_i}, \eta_{i,s}, PK_A\}$ to terminal u_i. (Note: for any two attributes $a_{i,k}$ and $a_{j,l}$ of different terminal members u_i and $u_j(i \ne j)$, if $a_{i,k} = a_{j,l}$, then $t_{i,k} = t_{j,l}$.)

5) After receiving the message $\{(T_{i,1}, T_{i,2}, ..., T_{i,r}), \mathrm{H}_{T_i}, \eta_{i,s}, PK_A\}$ from the Cloud, terminal u_i calculates the hash values $\mathrm{H}'_{T_i} = H_2(T_{i,1}||T_{i,2}||...||T_{i,r})$ and $\varepsilon_i = \lambda_i^{-1}T_{i1} + \lambda_i^{-1}T_{i,2} + ... + \lambda_i^{-1}T_{i,r} = (t_{i,1} + t_{i,2} + ... + t_{i,r})g_1$. u_i verifies that the attribute weights have not been tampered with during the transfer process by checking whether equation $\mathrm{H}'_{T_i} = \mathrm{H}_{T_i}$ holds. The identity of Cloud is verified by checking whether the signature $e(\eta_{i,s}, PK_A) = e(\varepsilon_i, g_1)$ of the Cloud holds. If the verification does not pass, the registration fails and terminal u_i broadcasts that the Cloud is a fake Cloud. If the verification passes, the terminal u_i obtains the attribute key set $\{K_{i,1}, K_{i,2}, ..., K_{i,r}\}$ by calculating the attribute key $K_{i,1} = \lambda_i^{-1}T_{i,1} = t_{i,1}g_1, K_{i,2} = \lambda_i^{-1}T_{i,2} = t_{i,2}g_1, ..., K_{i,r} = \lambda_i^{-1}T_{i,r} = t_{i,r}g_1$ and signature $sig_{u_i} = sk_{u_i}^{-1}\varepsilon_i$ corresponding to each attribute. Then u_i sends message $\{sig_{u_i}, pk_{u_i}\}$ to the Cloud.

6) After the Cloud receives the message $\{sig_{u_i}, pk_{u_i}\}$ from u_i, it compares the received terminal public key with the public key in the terminal authentication information table. (The public key in the table is obtained in the key self-certification process.) If the comparison is successful then the correctness of the terminal u_i identity information is further verified by the equation $e(sig_{u_i}, pk_{u_i}) = e(PK_A, \eta_{i,s})$. When the terminal u_i authentication fails, the registration is rejected, and when the authentication passes, the u_i attribute weights and other information are sent to ES.

Steps 1)-6) above are shown in Table 3.

Through the above steps, terminal $u_i(1 \le i \le n)$ is authenticated by the encrypted attributes and receives the attribute weights for each attribute. The Cloud records the attribute weight information in the terminal authentication information table (Table 2) and shares the table with ES for querying user's rights and resource access rights.

Table 3 Steps of terminal attribute authentication.

Cloud		$u_i(1 \le i \le n)$
		Select: $\lambda_i \in \mathbb{Z}_q^*$. Calculate: $\{(\lambda_i g_1, a_{i,1}\lambda_i g_1, a_{i,2}\lambda_i g_1, ..., a_{i,r}\lambda_i g_1), \mathrm{as}_{u_i} = H_2(a_{i,1}\lambda_i g_1 \| a_{i,2}\lambda_i g_1 \| ... \| a_{i,r}\lambda_i g_1), \beta_i = (a_{i,1} + a_{i,2} + ... + a_{i,r}) sk_{u_i}^{-1} \lambda_i g_1\}$.
	$\xleftarrow{\{(\lambda_i g_1, a_{i,1}\lambda_i g_1, a_{i,2}\lambda_i g_1, ..., a_{i,r}\lambda_i g_1), \mathrm{as}_{u_i}, \beta_i, pk_{u_i}\}}$	
Calculate: $\{\gamma_i = a_{i,1}\lambda_i g_1 + a_{i,2}\lambda_i g_1 + ... + a_{i,r}\lambda_i g_1, \mathrm{as}'_{u_i} = H_2(a_{i,1}\lambda_i g_1 \| a_{i,2}\lambda_i g_1 \| ... \| a_{i,r}\lambda_i g_1)\}$. Verify: $\{\mathrm{as}'_{u_i} \stackrel{?}{=} \mathrm{as}_{u_i}, e(\beta_i, pk_{u_i}) \stackrel{?}{=} e(\gamma_i, g_1)\}$. Calculate: equation (3). Verify: equation (4). Calculate: $\{(T_{i,0} = \lambda_i g_1, T_{i,1} = t_{i,1} T_{i,0}, T_{i,2} = t_{i,2} T_{i,0}, ..., T_{i,r} = t_{i,r} T_{i,0}), \mathrm{H}_{T_i} = H_2(T_{i,1} \| T_{i,2} \| ... \| T_{i,r}), \eta_{i,s} = SK_A^{-1}(t_{i,1} + t_{i,2} + ... + t_{i,r}) g_1\}$.		
	$\xrightarrow{\{(T_{i,1}, T_{i,2}, ..., T_{i,r}), \mathrm{H}_{T_i}, \eta_{i,s}, PK_A\}}$	
		Calculate: $\{\mathrm{H}'_{T_i} = H_2(T_{i,1} \| T_{i,2} \| ... \| T_{i,r}), \varepsilon_i = \lambda_i^{-1} T_{i1} + \lambda_i^{-1} T_{i,2} + ... + \lambda_i^{-1} T_{i,r} = (t_{i,1} + t_{i,2} + ... + t_{i,r}) g_1\}$. Verify: $\{\mathrm{H}'_{T_i} \stackrel{?}{=} \mathrm{H}_{T_i}, e(\eta_{i,s}, PK_A) \stackrel{?}{=} e(\varepsilon_i, g_1)\}$. Calculate: $\{(K_{i,1} = \lambda_i^{-1} T_{i,1}, K_{i,2} = \lambda_i^{-1} T_{i,2}, ..., K_{i,r} = \lambda_i^{-1} T_{i,r} = t_{i,r} g_1), sig_{u_i} = sk_{u_i}^{-1} \varepsilon_i\}$.
	$\xleftarrow{\{sig_{u_i}, pk_{u_i}\}}$	
Verify: $e(sig_{u_i}, pk_{u_i}) \stackrel{?}{=} e(PK_A, \eta_{i,s})$.		
Successful authentication.		

5.4 | Solution for secure data sharing

5.4.1 | Key exchange and data encryption storage

The brief encryption process is as follows.

Encryption: For any endpoint u_i that wants to share resources, first determine the scope of the sharing. Suppose u_i wants to share data m to terminals that have attribute combination a_1, a_2 or a_1, a_3, then compute the attribute weights v_1, v_2, corresponding to these two combinations respectively, and randomly select the encryption key r to encrypt the resource into the ciphertext Cm. The encryption function is $f_{Cm} = (x - v_1)(x - v_2) + r$. After expanding the encryption function, the coefficients are uploaded to the secure data sharing platform.

Each legitimate terminal in the system can store its data on the cloud server for data sharing. Confidential data need to be encrypted before uploading. The access rights to cryptographic resources are set by the data provider according to the level of confidentiality. Only members with the same or higher access rights to the resource can download and decrypt the cryptographic resource.

A user can define multiple access policies. Figure 3 represents that user u_j defines access policies $\Gamma u_{j,1}, \Gamma u_{j,2}, \cdots, \Gamma u_{j,k}$ for each of owned resources $m_{1,u_j}, m_{2,u_j}, \cdots, m_{k,u_j}$ in order to satisfy different users with different resources. Only users who match

the authorization set can access the resource. Each access policy can contain multiple combinations, and a resource viewer only needs to satisfy one of the permission combinations to be recognized as a user with that resource's authorization set.

Suppose that the data provider $u_j(1 \le j \le n)$ wants to share its resource to terminals with specific attributes. u_j owns the set of attributes $attr_j = (a_{j,1}, a_{j,2}, ..., a_{j,r})$ and the corresponding set of attribute authority values $\{T_{j,1}, T_{j,2}, ..., T_{j,r}\}$. Before u_j shares its data, it encrypts the shared data, and then uploads the shared ciphertext resources to the Edge Server. The procedure of calculating the key and storing the ciphertext is shown in Figure 4, and the steps are as follows:

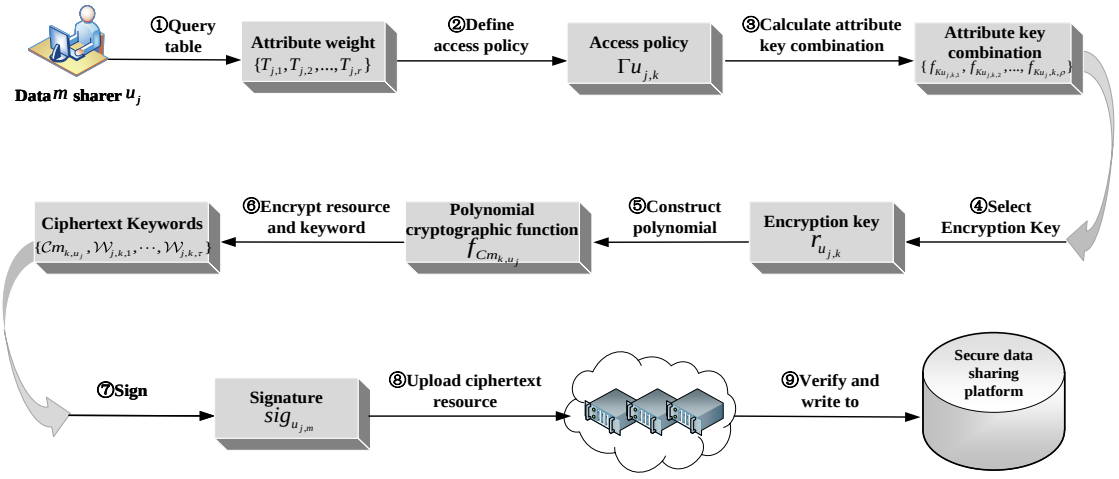

Figure 4 Data storage process.

1) u_j queries the terminal authentication information (Table 2) and set the attribute weights for accessing this resource. Assume that u_j sets the attribute weight set for accessing data $m_{k,u_j} \in \mathcal{M}$ ($\mathcal{M}$ is the plaintext space) as $\{T_{j,1}, ..., T_{j,r}\}$. The set of keywords corresponding to this data is $\{w_{j,k,1}, w_{j,k,2}, ..., w_{j,k,\tau}\}$. The access control policy $\Gamma u_{j,k}$ has $\rho(1 \le \rho \le r)$ combinations of attribute weights to access the data. (where k denotes the access policy for the kth shared data of u_j.) The combination of attribute weights is : $f_{Tu_{i,k,1}}, f_{Tu_{i,k,2}}, ..., f_{Tu_i,k,\rho}$. According to the combination of ρ attribute weights, u_j calculates the combination of attribute keys $f_{Ku_{j,k,1}}, f_{Ku_{j,k,2}}, ..., f_{Ku_j,k,\rho}$ corresponding to each combination of attribute weights respectively.

For example: The attribute permission policy u_j set to access data m_{k,u_j} is $\Gamma u_{j,k} = (T_{j,1}\ or\ T_{j,2})\ and\ T_{j,3} and\ (2\ of (T_{j,4}, T_{j,5}, T_{j,6}))$. According to the access control policy (Figure 5), there are six attribute weight combinations for accessing this data: $(T_{j,1}\ and\ T_{j,3}\ and\ T_{j,4}\ and\ T_{j,5})$, $(T_{j,1}\ and\ T_{j,3}\ and\ T_{j,4}\ and\ T_{j,6})$, $(T_{j,1}\ and\ T_{j,3}\ and\ T_{j,5}\ and\ T_{j,6})$, $(T_{j,2}\ and\ T_{j,3}\ and\ T_{j,4}\ and\ T_{j,5})$, $(T_{j,2}\ and\ T_{j,3}\ and\ T_{j,4}\ and\ T_{j,6})$ and $(T_{j,2}\ and\ T_{j,3}\ and\ T_{j,5}\ and\ T_{j,6})$. The combinations of the six attribute weights are:

$$
\begin{aligned}
f_{Tu_{j,k,1}} &= (T_{j,1}\ and\ T_{j,3}\ and\ T_{j,4}\ and\ T_{j,5}) \\
&= T_{j,1} + T_{j,3} + T_{j,4} + T_{j,5} \\
&= \lambda_j t_{j,1} g_1 + \lambda_j t_{j,3} g_1 + \lambda_j t_{j,4} g_1 + \lambda_j t_{j,5} g_1,
\end{aligned}
$$

$$
\begin{aligned}
f_{Tu_{j,k,2}} &= (T_{j,1}\ and\ T_{j,3}\ and\ T_{j,4}\ and\ T_{j,6}) \\
&= T_{j,1} + T_{j,3} + T_{j,4} + T_{j,6} \\
&= \lambda_j t_{j,1} g_1 + \lambda_j t_{j,3} g_1 + \lambda_j t_{j,4} g_1 + \lambda_j t_{j,6} g_1,
\end{aligned}
$$

$$\cdots$$

$$
\begin{aligned}
f_{Tu_{j,k,6}} &= (T_{j,2}\ and\ T_{j,3}\ and\ T_{j,5}\ and\ T_{j,6}) \\
&= T_{j,2} + T_{j,3} + T_{j,5} + T_{j,6} \\
&= \lambda_j t_{j,2} g_1 + \lambda_j t_{j,3} g_1 + \lambda_j t_{j,5} g_1 + \lambda_j t_{j,6} g_1.
\end{aligned}
$$

The attribute key combination corresponding to the combination of attribute weights is denoted as:

$$f_{Ku_{j,k,1}} = (K_{j,1}\ and\ K_{j,3}\ and\ K_{j,4}\ and\ K_{j,5}) = t_{j,1}g_1 + t_{j,3}g_1 + t_{j,4}g_1 + t_{j,5}g_1,$$

$$f_{Ku_{j,k,2}} = (K_{j,1}\ and\ K_{j,3}\ and K_{j,4}\ and\ K_{j,6}) = t_{j,1}g_1 + t_{j,3}g_1 + t_{j,4}g_1 + t_{j,6}g_1,$$

$$\cdots$$

$$f_{Ku_{j,k,6}} = (K_{j,2}\ and\ K_{j,3}\ and\ K_{j,5}\ and\ K_{j,6}) = t_{j,2}g_1 + t_{j,3}g_1 + t_{j,5}g_1 + t_{j,6}g_1.$$

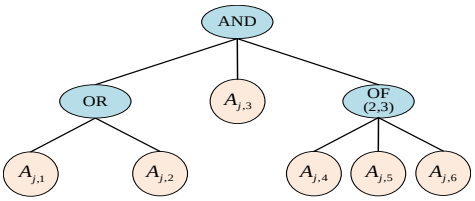

Figure 5 Resource m_{k,u_j} access policy tree.

2) u_j chooses a random number $r_{u_{j,k}} \in \mathbb{Z}_q^*$, calculates $v_{j,k,1} = H_2(f_{Ku_{j,k,1}})$, $v_{j,k,2} = H_2(f_{Ku_{j,k,2}}), \cdots, v_{j,k,\rho} = H_2(f_{Ku_{j,k,\rho}})$, and constructs a polynomial encryption function $f_{Cm_{k,u_j}} = (x - v_{j,k,\rho})(x - v_{j,k,\rho-1}) \cdots (x - v_{j.k.1}) + r_{u_{j,k}}$. Assume that the expanded polynomial is $f_{Cm_{k,u_j}} = x^{\rho} - b_{j,k,\rho-1}x^{\rho-1} ... - b_{j,k,1}x^1 - b_{j,k,0}$. u_j computes the ciphertext $Cm_{k,u_j} = m_{k,u_j} \oplus r_{u_{j,k}}$ and the set of ciphertext keywords $\{\mathcal{W}_{j,k,1} = w_{j,k,1} \oplus r_{u_{j,k}}, \mathcal{W}_{j,k,2} = w_{j,k,2} \oplus r_{u_{i,k}}, \cdots, \mathcal{W}_{j,k,\tau} = w_{j,k,\tau} \oplus r_{u_{j,k}}\}$ of the shared data.

3) u_j calculates the signature $sig_{u_{j,m}} = sk_{u_j}^{-1} H_1(Cm_{k,u_j}||\mathcal{W}_{j,k,1}||\mathcal{W}_{j,k,2}||...||\ \mathcal{W}_{j,k,\tau}||b_{j,k,\rho-1}||b_{j,k,\rho-2}||...||b_{j,k,1}||b_{j,k,0}||\Gamma u_{j,k})g_1$ of the message, and sends the message $\{Cm_{k,u_j}, (\mathcal{W}_{j,k,1}, \mathcal{W}_{j,k,2}, ..., \mathcal{W}_{j,k,\tau}), (b_{j,k,\rho-1}, b_{j,k,\rho-2}, ..., b_{j,k,1}, b_{j,k,0}), \Gamma u_{j,k}, sig_{u_{j,m}}\}$ to ES.

4) After ES receives the message sent by u_j, it calculates $hm_{u_j} = H_1(Cm_{k,u_j}\ ||\mathcal{W}_{j,k,1}||\mathcal{W}_{j,k,2}||...||\mathcal{W}_{j,k,\tau}||b_{j,k,\rho-1}||b_{j,k,\rho-2}||...$ $||b_{j,k,1}||b_{j,k,0}||\Gamma u_{j,k})g_1$. The identity of u_j and the messages it sends are verified by equation $e(sig_{u_{j,m}}, pk_{u_j}) = e(hm_{u_j}, g_1)$. If the authentication passes, ES returns a successful storage message to the user and writes the information to the data sharing platform (as shown in Table 4). If the authentication fails, the ES returns "Storage failed".

Table 4 Secure data sharing platform.

Terminal	u_1	u_2	**...**	u_n
Validity	yes	yes	...	yes
Public Key	pk_{u_1}	pk_{u_2}	...	pk_{u_n}
Shared Ciphertext Data Hash	$H_1(Cm_{k,u_1})$	$H_1(Cm_{k,u_2})$	...	$H_1(Cm_{k,u_n})$
Ciphertext Keyword	$\mathcal{W}_{1,k,1}, \mathcal{W}_{1,k,2}, ..., \mathcal{W}_{1,k,\tau})$	$\mathcal{W}_{2,k,1}, \mathcal{W}_{2,k,2}, ..., \mathcal{W}_{2,k,\tau})$	...	$\mathcal{W}_{n,k,1}, \mathcal{W}_{n,k,2}, ..., \mathcal{W}_{n,k,\tau})$
Encryption function coefficient	$(b_{1,k,\rho-1}, b_{1,k,\rho-2}, ..., b_{1,k,0})$	$(b_{2,k,\rho-1}, b_{2,k,\rho-2}, ..., b_{2,k,0})$	...	$(b_{n,k,\rho-1}, b_{n,k,\rho-2}, ..., b_{n,k,0})$
Attribute weight	$\{T_{1,1}, T_{1,2}, T_{1,3}\}$	$\{T_{2,1}, T_{2,2}, ..., T_{2,9}\}$	...	$\{T_{n,1}, T_{n,2}, ..., T_{n,r}\}$
Access Control Policy	$\Gamma u_{1,k}$	$\Gamma u_{2,k}$	...	$\Gamma u_{n,k}$

5.4.2 | Download and access to shared resources

The brief decryption process is as follows.

Decryption: The terminal u_j that owns the attributes a_1, a_2 is eligible for resource m access. Based on the owned attributes, u_j can calculate the attribute key combination v_1. Then u_j constructs the function based on the coefficients of the encryption function on the secure data sharing platform. The decryption key r is obtained by substituting v_1 into the equation.

If any data requestor $u_i (1 \leq i \leq n)$ in the system needs to search and download the shared data on the cloud platform, the process required to access the data is shown in Figure 6. The details are as follows:

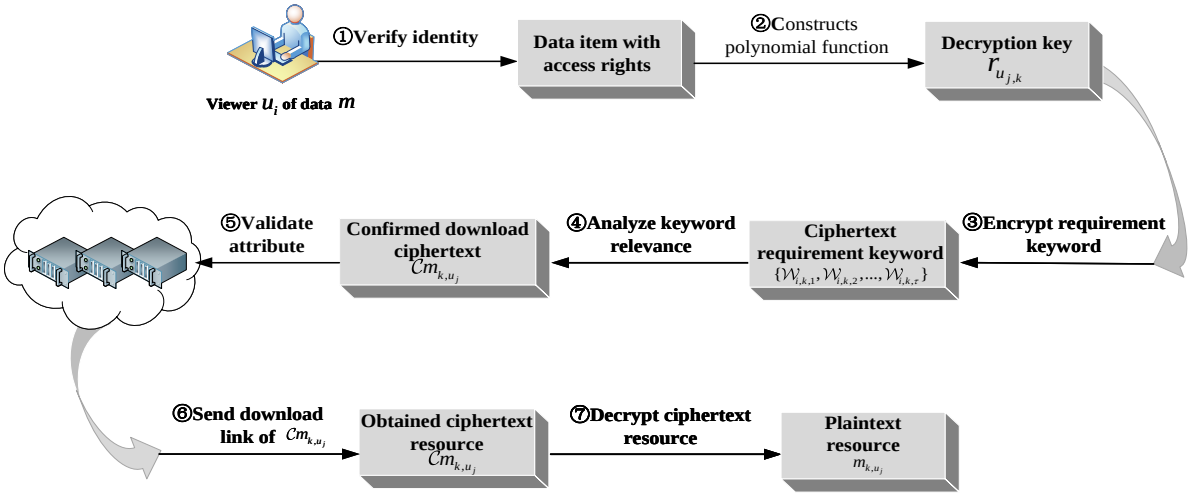

Figure 6 Data access process.

1) Before obtaining access rights to the data items, u_i needs to prove to the ES that it is a legitimate terminal. u_i finds data items with access rights based on the set of attribute weights and access control policies of shared data on the data sharing platform. The polynomial function is constructed through the cryptographic function coefficients in the data items. With the access control policy and attribute weights on the platform, the attribute key combination and its hash value can be calculated. By substituting the hash into the polynomial function, the decryption key of the shared data can be calculated by u_i.

For example, u_i queries the data sharing platform and finds that it has access to data Cm_{k,u_j} (which was uploaded by u_j). u_i constructs a polynomial function $f_{Cm_{k,u_j}} = x^{\rho} - b_{j,k,\rho-1}x^{\rho-1} ... - b_{j,k,1}x^1 - b_{j,k,0}$ based on the encryption function coefficients $(b_{j,k,\rho-1}, b_{j,k,\rho-2}, ..., b_{j,k,1}, b_{j,k,0})$ of Cm_{k,u_j}. Based on the attribute weight set $\{T_{j,1}, ..., T_{j,r}\}$ of the shared data and the access control policy $\Gamma u_{j,k}$, u_i can calculate the corresponding attribute key combination $f_{Ku_{i,2}} = (K_{i,1}\ and\ K_{i,3}\ and\ K_{i,4}\ and\ K_{i,6}) = t_{i,1}g_1 + t_{i,3}g_1 + t_{i,4}g_1 + t_{i,6}g_1$. (Assume that u_i only satisfies this access control policy.) The hash values $v_{i,k,2} = H_2(f_{Ku_{i,k,2}})$ and $f_{Cm_{k,u_j}}(v_{i,k,2}) = r_{u_{j,k}}$ can be computed by u_i. (Note: Because $K_{i,1} = K_{j,1}$, $K_{i,3} = K_{j,3}$, $K_{i,4} = K_{j,4}$, $K_{i,6} = K_{j,6}$, therefore $v_{i,k,2} = v_{j,k,2}$.) That is, u_i can calculate the decryption key $r_{u_{j,k}}$ for the shared data.

2) Assume that the set of plaintext keywords for the data demanded by u_i is $\{w_{i,k,1}, w_{i,k,2}, ..., w_{i,k,\tau}\}$. By using the decryption key $r_{u_{j,k}}$ of the shared data, u_i calculates the ciphertext keyword $\{\mathcal{W}_{i,k,1} = w_{i,k,1} \oplus r_{u_{j,k}}, \mathcal{W}_{i,k,2} = w_{i,k,2} \oplus r_{u_{i,k}}, ..., \mathcal{W}_{i,k,\tau} = w_{i,k,\tau} \oplus r_{u_{j,k}}\}$ of its searched ciphertext data, and compares it with the keyword $(\mathcal{W}_{j,k,1}, \mathcal{W}_{j,k,2}, ..., \mathcal{W}_{j,k,\tau})$ of Cm_{k,u_j}. Data relevance is divided according to the number of keywords corresponding. If most of the ciphertext keywords are the same, then Cm_{k,u_j} can be downloaded by u_i.

3) The ciphertext resource can only be downloaded by u_i after passing the attribute permission check of ES. After calculating the hash value $\mathrm{H}'_{T_i} = H_2(T_{i,1}||T_{i,2}||...||T_{i,r})$ of the attribute weights and its signature $sig_{u_{i,m}} = sk_{u_i}^{-1}\mathrm{H}'_{T_i}g_1$, the message $\{H_1(Cm_{k,u_j}), (T_{i,1}, T_{i,2}, ..., T_{i,r}), \mathrm{H}'_{T_i}, sig_{u_{i,m}}, u_i, u_j\}$ is sent by u_i to ES. After ES receives the message, it calculates $\mathrm{H}''_{T_i} = H_2(T_{i,1}||T_{i,2}||...||T_{i,r})$ based on the attribute weights in the terminal authentication information table, and checks whether H''_{T_i} and H'_{T_i} are equal. If they are equal, then the identity of u_i is verified by computing the equation $e(sig_{u_{i,m}}, pk_{u_i}) = e(\mathrm{H}'_{T_i}, g_1)$. The Cloud compares the received attribute weight information $(T_{i,1}, T_{i,2}, ..., T_{i,r})$ with u_i' attribute weights in the data sharing platform. If the comparison passes, the ciphertext data link is encrypted and sent to u_i. According to the link Cm_{k,u_j} can be downloaded by u_i, and the ciphertext hash value can be calculated. If the calculated ciphertext hash is the same as that on the data sharing platform, Cm_{k,u_j} can be decrypted into plaintext $m_{k,u_j} = Cm_{k,u_j} \oplus r_{u_{j,k}}$ by the $r_{u_{j,k}}$.

5.4.3 | Application cases

End-Edge-Cloud Vehicles, road-side units (RSU), and the Cloud are needed to work together in the Internet of vehicles (IoV) application scenario. The computational power of vehicles is very limited and cannot load all the computational workload. RSU is only capable of undertaking a small number of calculations. And the Cloud has many servers that can take up a larger amount of computing. The distance between vehicles and the Cloud is distant and the communication delay is long. By converting vehicle-to-cloud communication into vehicle-to-RSU and cloud-to-RSU communication, the communication delay can be greatly reduced.

End-Cloud In networks with End-Cloud architecture, such as Amazon Cloud Services, a large number of calculations performed at the terminal may take longer to compute due to limited computing power. If the computation is moved to the cloud, it requires the terminal to communicate with the remote cloud over long distances, which will increase communication latency. Therefore, we recommend edge-cloud collaborative architecture for such cases. As shown in Figure 7. Terminals can move a lot of computation to Edge servers closer to them, which will greatly reduce communication latency as well as alleviate the lack of computing power.

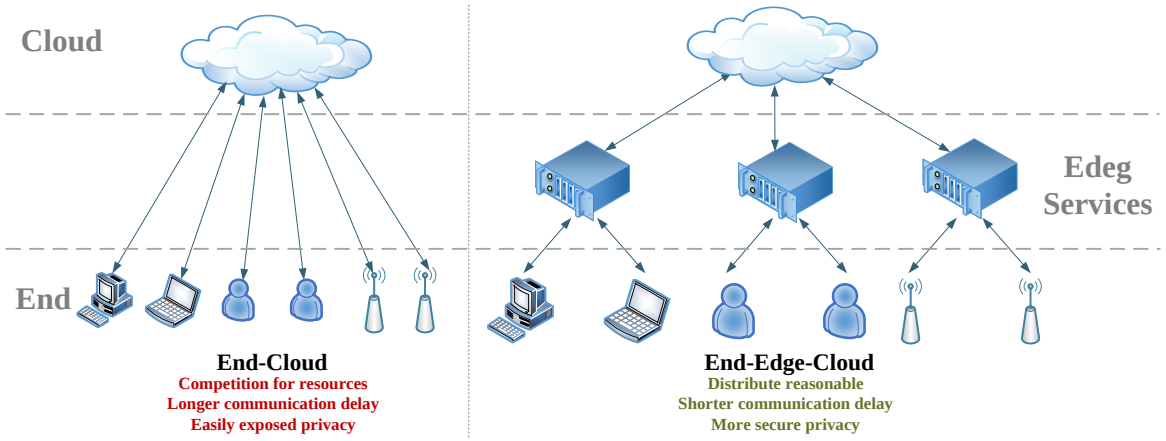

Figure 7 End-Cloud architecture and End-Edge-Cloud architecture.

6 | MODEL SECURITY ANALYSIS

Correctness and security analysis are essential for a full evaluation of the model. In this section, the correctness of the above scheme is first analyzed, and then the security of the model is explained.

6.1 | Correctness analysis

Theorem 1 Any honest terminal u_{honest} can be authenticated by the Cloud. If Equation $e(\mathrm{Term}_{u_{honest}}, \mathrm{H}_{u_{honest}}) = e(pk_{u_{honest}}, g_1)$ holds, then terminal u_{honest} can be recognized by the Cloud as a legitimate user, and its public/private key pair $(sk_{u_{honest}}, pk_{u_{honest}})$ will be legitimized.

Proof Let's assume that Cloud receives the message sent by terminal u_{honest} as $\{PK_{u_{honest}}, pk_{u_{honest}}\}$, according to the received message $\mathrm{Term}_{u_{honest}} = SK^{-1}_{u_{honest}} PK_A$ can be calculated. Since the Cloud has the ID information of the terminals in the system, it can calculate $\mathrm{H}_{u_{honest}} = H_1(id_{u_{honest}})g_1$. By the property of bilinear mapping we have:

$$\begin{aligned} & e(\mathrm{Term}_{u_{honest}}, \mathrm{H}_{u_{honest}}) \\ &= e(SK^{-1}_{u_{honest}} PK_A, H_1(id_{u_{honest}})g_1) \\ &= e(s_{u_{honest}} g_1, H_1(id_{u_{honest}})g_1) \\ &= e(H_1(id_{u_{honest}})s_{u_{honest}} g_1, g_1) \\ &= e(pk_{u_{honest}}, g_1) \end{aligned}$$

Therefore, the above equation can be used to verify the identity of the terminal u_{honest} and to obtain a mutually agreed public/private key pair $(sk_{u_{honest}}, pk_{u_{honest}})$.

Theorem 2 Legitimate user u_{legal} can verify his attributes $attr_{u_{legal}} = \{a_{legal,1}, a_{legal,2}, ..., a_{legal,r}\}(1 \le r \le R)$ with the Cloud. First, the identity of u_{legal} needs to be determined by the equation $e(\beta_{legal}, pk_{u_{legal}}) = e(\gamma_{legal}, g_1)$. If this equation holds, the corresponding attribute weights are returned by the Cloud to u_{legal}. Then u_{legal} verifies the identity of Cloud in the reverse direction by equation $e(\eta_{legal,s}, g_1) = e(\varepsilon_{legal}, PK_A)$. Finally, if equation $e(sig_{u_{legal}}, pk_{u_{legal}}) = e(PK_A, \eta_{legal,s})$ holds, the attribute weights and other information will be saved to the cloud service platform.

Proof Because $\beta_{legal} = (a_{legal,1} + a_{legal,2} + ... + a_{legal,r})sk_{u_{legal}}^{-1}\lambda_{legal}g_1$ and $\gamma_{legal} = a_{legal,1}\lambda_{legal}g_1 + a_{legal,2}\lambda_{legal}g_1 + ... + a_{legal,r}\lambda_{legal}g_1$, by the property of bilinear mapping we get:

$$\begin{aligned} & e(\beta_{legal}, pk_{u_{legal}}) \\ &= e((a_{legal,1} + a_{legal,2} + ... + a_{legal,r})sk_{u_{legal}}^{-1}\lambda_{legal}g_1, sk_{u_{legal}}g_1) \\ &= e((a_{legal,1} + a_{legal,2} + ... + a_{legal,r})\lambda_{legal}g_1, g_1)^{sk_{u_{legal}}^{-1}sk_{u_{legal}}} \\ &= e(\gamma_{legal}, g_1) \end{aligned}$$

Because $\varepsilon_{legal} = (t_{legal,1} + t_{legal,2} + ... + t_{legal,r})g_1$ and $\eta_{legal,s} = SK_A^{-1}(t_{legal,1} + t_{legal,2} + ... + t_{legal,r})g_1$, by the property of bilinear mapping we get:

$$\begin{aligned} & e(\eta_{legal,s}, PK_A) \\ &= e(SK_A^{-1}(t_{legal,1} + t_{legal,2} + ... + t_{legal,r})g_1, SK_A g_1) \\ &= e((t_{legal,1} + t_{legal,2} + ... + t_{legal,r})g_1, g_1) \\ &= e(\varepsilon_{legal}, g_1) \end{aligned}$$

Because $sig_{u_{legal}} = sk_{u_{legal}}^{-1}\varepsilon_{legal}$, by the property of bilinear mapping we get:

$$\begin{aligned} & e(sig_{u_{legal}}, pk_{u_{legal}}) \\ &= e(sk_{u_{legal}}^{-1}\varepsilon_{legal}, sk_{u_{legal}}g_1) \\ &= e(sk_{u_{legal}}^{-1}(t_{legal,1} + t_{legal,2} + ... + t_{legal,r})g_1, sk_{u_{legal}}g_1) \\ &= e(PK_A, \eta_{legal,s}) \end{aligned}$$

As a result, legitimate users can be authenticated by the Cloud for the attribute, and receive the corresponding attribute privileges.

Theorem 3 User u_{sharer} can upload shared resources to the Cloud Server. ES checks equation $e(sig_{u_{sharer,m}}, pk_{u_{sharer}}) = e(hm_{u_{sharer}}, g_1)$. If the equation holds, the resource is shared successfully and stored encrypted in the data sharing platform.

Proof Since $sig_{u_{sharer,m}} = sk_{u_{sharer}}^{-1}H_1(Cm_{k,u_{sharer}}||\mathcal{W}_{sharer,k,1}||\mathcal{W}_{sharer,k,2}||...||\mathcal{W}_{sharer,k,\tau}||b_{sharer,k,\rho-1}||b_{sharer,k,\rho-2}||...||b_{sharer,k,1}||b_{sharer,k,0}||\Gamma u_{sharer,k})g_1$, $hm_{u_{sharer}} = H_1(Cm_{k,u_{sharer}}||\mathcal{W}_{sharer,k,1}||\mathcal{W}_{sharer,k,2}||...||\mathcal{W}_{sharer,k,\tau}||b_{sharer,k,\rho-1}||b_{sharer,k,\rho-2}||...||b_{sharer,k,1}||b_{sharer,k,0}||\Gamma u_{sharer,k})g_1$, it follows from the properties of bilinear mappings:

$$\begin{aligned} & e(sig_{u_{sharer,m}}, pk_{u_{sharer}}) \\ =& \mathrm{e}(sk_{u_{sharer}}^{-1}H_1(Cm_{k,u_{sharer}}||\mathcal{W}_{sharer,k,1}||\mathcal{W}_{sharer,k,2}||...||\mathcal{W}_{sharer,k,\tau}|| \\ & b_{sharer,k,\rho-1}||b_{sharer,k,\rho-2}||...||b_{sharer,k,1}||b_{sharer,k,0}||\Gamma u_{sharer,k})g_1, sk_{u_{sharer}}g_1) \\ =& \mathrm{e}(H_1(Cm_{k,u_{sharer}}||\mathcal{W}_{sharer,k,1}||\mathcal{W}_{sharer,k,2}||...||\mathcal{W}_{sharer,k,\tau}||b_{sharer,k,\rho-1}|| \\ & b_{sharer,k,\rho-2}||...||b_{sharer,k,1}||b_{sharer,k,0}||\Gamma u_{sharer,k})g_1, g_1) \\ =& e(hm_{u_{sharer}}, g_1) \end{aligned}$$

Therefore, legitimate users can upload shared resources to ES.

Theorem 4 Resources can be downloaded by users who meet the access policy. If the access policy $\Gamma u_{user,k}$ of resource $Cm_{k,u_{user}}$ is satisfied by u_{user}, then the user can calculate the decryption key of resource and obtain plaintext resource $m_{k,u_{user}}$. Equation $e(sig_{u_{user,m}}, pk_{u_{user}}) = e(\mathrm{H}'_{T_{user}}, g_1)$ verifies that the user is considered a legitimate user. Then the plaintext resource can be computed using the decryption key and the downloaded ciphertext resource.

Proof Since $sig_{u_{user,m}} = sk_{u_{user}}^{-1}\mathrm{H}'_{T_{user}}g_1$ and $\mathrm{H}'_{T_{user}} = H_2(T_{user,1}||T_{user,2}||...||T_{user,r})$, we get from the property of bilinear mapping:

$$\begin{aligned} & e(sig_{u_{user,m}}, pk_{u_{user}}) \\ &= e(sk_{u_{user}}^{-1}\mathrm{H}'_{T_{user}}g_1, sk_{u_{user}}g_1) \\ &= e(\mathrm{H}'_{T_{user}}, g_1) \end{aligned}$$

Because $Cm_{k,u_{user}} = m_{k,u_{user}} \oplus r_{u_{user,k}}$, then:

$$\begin{aligned} & Cm_{k,u_{user}} \oplus r_{u_{user,k}} \\ =& m_{k,u_{user}} \oplus r_{u_{user,k}} \oplus r_{u_{user,k}} \\ =& m_{k,u_{user}} \end{aligned}$$

Therefore, users who have access rights to the resource can download and decrypt the resource.

6.2 | Security analysis

Theorem 5 Resources with higher access privileges cannot be accessed by users with lower access privileges. For a resource Cm_{k,u_j} that requires access permission $\eta_{j,s}$, it cannot be accessed if user u_i has permission $\eta_{i,s}$ less than $\eta_{j,s}$, where $i, j \in [1, n]$.

Proof Assume that access to resource Cm_{k,u_j} requires attribute weights $(T_{j,1}, T_{j,2}, ..., T_{j,k})$. u_i has an attribute with weight $(T_{i,1}, T_{i,2}, ..., T_{i,k-1})$ and wants to access resource Cm_{k,u_j}.

u_i calculates $(\lambda_i g_1, a_{i,1}\lambda_i g_1, a_{i,2}\lambda_i g_1, ..., a_{i,k-1}\lambda_i g_1)$, $\text{as}_{u_i} = H_2(a_{i,1}\lambda_i g_1 \,||a_{i,2}\, \lambda_i g_1||...||a_{i,k-1}\lambda_i g_1)$ and $\beta_i = (a_{i,1} + a_{i,2} + ... + a_{i,k-1})sk_{u_i}^{-1}\lambda_i g_1$. After the properties of u_i are verified by the Cloud, their attribute weights are calculated and assigned to authority levels S. The access policy for Cm_{k,u_j} is $\Gamma u_{j,k}$ (Assuming only this access control policy is available, the corresponding attribute key is $f_{Ku_{j,k}}$.). Polynomial functions cannot be constructed by u_i because the cryptographic function coefficients are not available from the shared platform. And using the set of attribute weights $(T_{i,1}, T_{i,2}, ..., T_{i,k-1})$ and access control policy $\Gamma u_{i,k}$, $f_{Ku_{j,k}}$ cannot be computed by u_i.

Theorem 6 The model is resistant to man-in-the-middle attacks. In the process of passing messages among entities, attacker maybe intercept the message and send altered message to original recipient. The tampered information can be identified by the original recipient. Attacker cannot recover private data by stealing the contents of the communication.

Proof Suppose the message $\{(\lambda_i g_1, a_{i,1}\lambda_i g_1, a_{i,2}\lambda_i g_1, ..., a_{i,r}\lambda_i g_1), \text{as}_{u_i}, \beta_i, pk_{u_i}\}$ sent by u_i to the Cloud is intercepted by an attacker, who removes $a_{i,r}\lambda_i g_1$ from it and sends the altered message to the Cloud. The Cloud calculates $\gamma_i = a_{i,1}\lambda_i g_1 + a_{i,2}\lambda_i g_1 + ... + a_{i,r-1}\lambda_i g_1$ and $\text{as}'_{u_i} = H_2(a_{i,1}\lambda_i g_1||a_{i,2}\lambda_i g_1||...||\, a_{i,r-1}\lambda_i g_1)$ after receiving the message and compares whether as'_{u_i} and as_{u_i} are equal. Equation $e(\beta_i, pk_{u_i}) = e(\gamma_i, g_1)$ also needs to be verified. Since $\text{as}_{u_i} = H_2(a_{i,1}\lambda_i g_1||a_{i,2}\lambda_i g_1\, ||...||a_{i,r}\lambda_i g_1) \neq \text{as}'_{u_i}$, and:

$$\begin{aligned} & e(\beta_i, pk_{u_i}) \\ =\ & e((a_{i,1} + a_{i,2} + ... + a_{i,r})sk_{u_i}^{-1}\lambda_i g_1, sk_{u_i} g_1) \\ =\ & e(g_1, g_1)^{(a_{i,1}+a_{i,2}+...+a_{i,r})sk_{u_i}^{-1}\lambda_i sk_{u_i}} \\ =\ & e(g_1, g_1)^{(a_{i,1}+a_{i,2}+...+a_{i,r})\lambda_i} \\ \neq\ & e(\gamma_i, g_1) \end{aligned}$$

Therefore, the tampered information can be recognized by the Cloud and other requests from u_i can be denied to achieve anti-man-in-the-middle attack.

Theorem 7 User u_i with attribute $\{a_{i,1}, a_{i,2}, ..., a_{i,r}\}(1 \leq r \leq R)$ can only obtain the attribute key $(K_{i,1}\ ,\ K_{i,2}\ , ...,\ K_{i,r})$ corresponding to these r attributes. The attribute key corresponding to the attribute that does not have can not be obtained by u_i.

Proof Suppose the resource user u_d wants to decrypt requires that the attribute $a_{d,k}$ must be present. u_d has no attribute $a_{d,k}$ and user u_i has attribute $a_{i,k}$. (Where $a_{d,k} = a_{i,k}$.)

(1) If information $T_{i,k} = t_{i,k}T_{i,0}$ sent by the Cloud to user u_i is successfully intercepted by user u_d, where $T_{i,0} = \lambda_i g_1$. Need to have λ_i for the u_d to calculate the attribute key $K_{i,k} = \lambda_i^{-1}T_{i,k}$. λ_i is a private parameter of u_i (not known to any terminal or device other than the user itself), so the attribute key corresponding to attribute $a_{d,k}$ cannot be directly calculated by u_d. That is, the decryption key of the data cannot be calculated.

(2) If u_d also intercepts the information $\{\lambda_i g_1, a_{i,k}\lambda_i g_1\}$ sent to the Cloud by u_i. At this time, it seems that u_d can combine $\lambda_i g_1$ and $T_{i,k} = t_{i,k}T_{i,0}$ to calculate the attribute key $K_{i,k}$. Let $\lambda_i g_1$ be ag_1, and let $T_{i,k}$ be abg_1, so $K_{i,k} = \lambda_i^{-1}T_{i,k}$=$(ab/a)g_1$. According to the assumption of Inv-CDHP, the attribute key $K_{i,k}$ is difficult to be computed by u_d. Therefore, the terminal cannot obtain the attribute key of unowned attributes and cannot participate in resource sharing with unknown attributes.

7 | MODEL PERFORMANCE ANALYSIS

Another important evaluation of a protocol is its performance. In this section, we compare and analyze the scheme proposed in [32-34] and this paper in terms of computational complexity and computational time overhead.

7.1 | Computational complexity analysis

Let T_h be a hash operation, T_{mul} be a scalar multiplication operation, $T_{\exp}$ be an exponential operation, T_{inv} be a modulo inverse operation, $T_{pa\text{-}ecc}$ be an point addition operation over elliptic curve, T_{sm-ecc} be a point multiplication operation over elliptic curve, and T_{bp} be a bilinear pairing operation. Because of the short time required for heterogeneous and additive operations compared to $T_{\exp}$ and T_{bp} calculations, they are neglected. Let there are n users participating in ciphertext resource sharing. For example, user u_i who uploads ciphertext resources has r attributes and ρ combinations of attribute weights.

The scheme proposed in this paper requires the computation of sk_{u_i}, PK_{u_i} and pk_{u_i} in the registration phase, which consists of one hash operation, one scalar multiplication and two addition operations on elliptic curves. In the key generation phase, it is necessary to calculate the intermediate parameters $(\lambda_i g_1, a_{i,1}\lambda_i g_1, a_{i,2}\lambda_i g_1, ..., a_{i,r}\lambda_i g_1)$, β_i and the signature as_{u_i}. After receiving the message from the Cloud, H'_{T_i} and ε_i are computed and equation $e(\eta_{i,s}, PK_A)=e(\varepsilon_i, g_1)$ is verified. Then $(K_{i,1}, K_{i,2}, ..., K_{i,r})$ and signature sig_{u_i} were calculated. Therefore, this stage uses 2 hash operations, 1 scalar multiplication operation, $2(r+1)$ modulo inverse operations, $(r+2)$ point multiplication operations over elliptic curve, and 2 bilinear pairing operations. In the encryption phase, the polynomial needs to be expanded and the signature $sig_{u_{j,m}}$ of the message needs to be computed, which requires 1 hash operation, $(\rho-1)$ scalar multiplication operations, 1 modulo inverse operation and 1 point multiplication operation over elliptic curve. In the decryption phase, the hash value of the attribute key combination is calculated and substituted into a polynomial function to derive the decryption key for the ciphertext data. H'_{T_i} and signature $sig_{u_{i,m}}$ need to be calculated when authenticating with ES. After the resource is downloaded, the ciphertext hash is calculated again and compared with the value of the data sharing platform to ensure that the ciphertext has not been tampered with. In this process, 3 hash operations, ρ exponential operations, 1 modulo inverse operation and 1 point multiplication operation over elliptic curve are used.

Table 5 Computational complexity of the four schemes.

	Registration stage	Key generation stage	Encryption stage	Decryption stage
Li et al.[32]	$(n+5)T_{\exp}+T_{bp}$	$(3+2r)T_{\exp}$	$(n+3+2r)T_{\exp}$	$(n+r)T_{\exp}+(n+2r+2)T_{bp}$
Deng et al.[33]	$T_{pa-ecc}+T_{bp}$	$(3r+4)T_{\exp}$	$(5r+3)T_{\exp}$	$rT_{\exp}+(3r+1)T_{bp}$
Xue et al.[34]	$(2r+4)T_{\exp}+T_{bp}$	$3rT_{\exp}+rT_{bp}$	$(4r+1)T_{\exp}+T_{bp}$	$3rT_{bp}$
Our protocol	$T_h+T_{mul}+2T_{sm-ecc}$	$2T_h+T_{mul}+2(r+1)T_{inv}+(r+2)T_{sm-ecc}+2T_{bp}$	$T_h+(\rho-1)T_{mul}+T_{inv}+T_{sm-ecc}$	$3T_h+\rho T_{\exp}+T_{inv}+T_{sm-ecc}$

The results of the analysis of the computational complexity of the four scenarios are shown in Table 5, where n denotes the number of terminals, r denotes the number of network attributes, and ρ denotes the number of attribute access policies. From an overall perspective, the sharing scheme proposed in this paper is the least computationally intensive, followed by Li et al.[32] and Deng et al.[33]. Xue et al.[34] is the most complex. In the registration phase, good performance results are achieved in Xue et al.[34] only when the user's attributes are sufficiently few,since the computational complexity is proportional to the number of attributes of the terminal. In the key generation stage, this paper uses a simpler heteroskedastic operation in the Cloud to authenticate the attributes of the terminal, which greatly reduces the computational complexity. The literature [32-34] continue to use $T_{\exp}$ and T_{bp} with high computational complexity. In the encryption stage, Deng et al.[33] has the most complicated calculation, where for each additional attribute, a corresponding increase of $5T_{\exp}$ is applied. Followed by Xue et al.[34] and Li et al.[32]. The scheme proposed in this paper is the easiest to compute, and the main computational comes from the expansion of the encryption polynomial. In the decryption phase, the model described in this paper still has the simplest computational degree, followed by Xue et al.[34] and Li et al.[32]. Deng et al.[33] has the most complex computations. It is mainly because Deng et al.[33] contains $rT_{\exp}$ as well as $(3r+1)T_{bp}$, which greatly increases the complexity of the calculation.

7.2 | Calculation time overhead analysis

To meet the application of resource-constrained scenarios, we choose to analyze the time overhead of each operation in a terminal environment with weak computing power. This paper uses the JPBC cryptographic library (JPBC-2.0.0) in the Java programming language and experiments on the Android terminal. The terminal configuration is described in Table 6. Table 7 shows the experimental data for the seven types of operations consuming longer time.

Table 6 Experimental configuration.

Device name	:	HONOR 60
Android version	:	12
Processor	:	Qualcomm Snapdragon 778G
RAM	:	8.0 GB + 2.0 GB (HONOR RAM Turbo)
Internal storage	:	256 GB
Kernel version	:	5.4.147-qgki-g45a99f875b89 android@localhost #1 Tue Jan 3 20:15:59 CST 2023

Table 7 Time consumption of various operations (unit: ms).

Operation type	**Time consumption**
T_{bp} Bilinear pairing operation	$T_{bp} \approx 12.0577$
$T_{\exp}$ Exponential operation	$T_{\exp} \approx 9.9311$
T_{sm-ecc} Point Multiplication operation over elliptic curve	$T_{sm-ecc} \approx 0.0605$
$T_{pa\text{-}ecc}$ Point addition operation over elliptic curve	$T_{pa\text{-}ecc} \approx 0.0552$
T_{inv} Modular-inversion operation	$T_{inv} \approx 0.0146$
T_{mul} Scalar multiplication operation	$T_{mul} \approx 0.0009$
T_h Hash operation	$T_h \approx 0.0005$

The total time spent on each operation is obtained by multiplying the time and number of times used for that operation. The sum of the time consumed by all operations is the total time consumed by the calculation. For the purpose of analysis, experiments are conducted for the ordered set of attributes $Attr$ with numbers 2, 4, 8 and 16. The number of attribute access policies ρ and terminals n grows linearly with the number of attributes. With the data in Tables 5 and 6, the time overheads of each of the four phases are compared and analyzed. The total time we show in the manuscript in Figures 8-11.

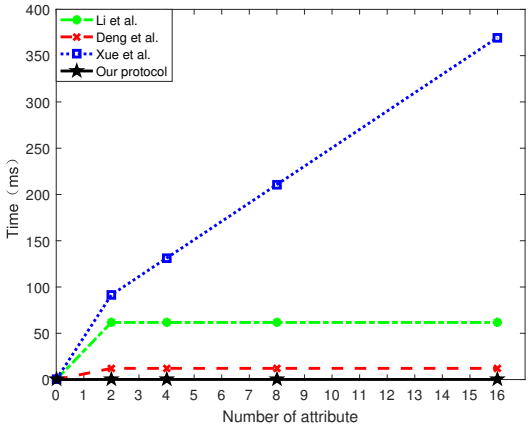

Figure 8 Comparison of computational time consumption of four schemes in the registration phase.

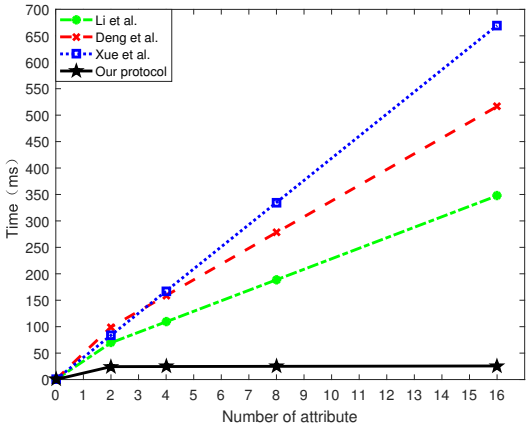

Figure 9 Comparison of computational time consumption of four schemes in the key generation phase.

As shown in Figure 8, the proposed scheme consumes the least amount of computation time during the registration phase, because only $2T_{sm-ecc}$, $1T_{mul}$ and $1T_h$ are used in our protocol. Followed by Deng et al.[33] and Li et al.[32]. Since Xue et al.[34] marks the groups of users by different master keys, $(2r+4)$ $T_{\exp}$ and 1 T_{bp} are consumed in the process. It does not have good performance in this phase compared to other schemes. And as the number of terminal attributes increases, longer computation time is required by Xue et al.[34].

Figure 9 shows the comparison of computation time of the four schemes in the key generation phase. In this phase, our scheme uses exclusive-OR to authenticate attributes of the terminal. The exclusive-OR operations in cryptography do not generate

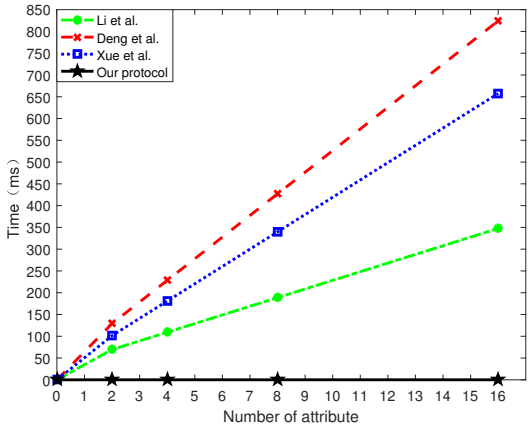

Figure 10 Comparison of computational time consumption of four schemes in the encryption phase.

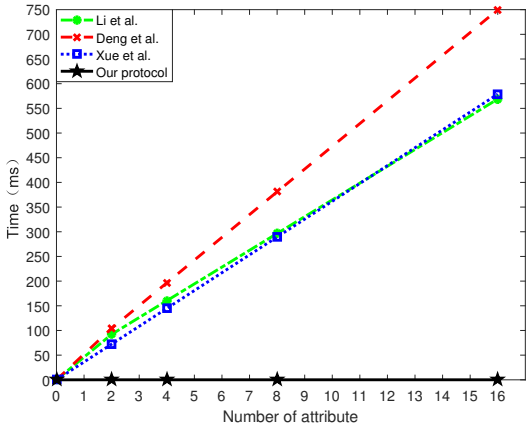

Figure 11 Comparison of computational time consumption of four schemes in the decryption phase.

rounding compared to the quadratic operations (addition, subtraction, multiplication and division), and the computations can be performed in parallel more efficiently among various parts. The received information is confirmed by $2T_{bp}$. Therefore, the calculation consumes the least amount of time. Li et al.[32] is in the middle of the calculation time compared to other schemes, requiring a total of $(3+2r)T_{\exp}$. When the number of attributes of the terminal is less than 4, Xue et al.[34] has less time consumption than Deng et al.[33]. When the number of attributes owned by the terminal is more than or equal to 4, the time consumption of Deng et al.[33] is better than Xue et al.[34]. The reason is that the number of operations in Xue et al.[34] is related to the number of attributes r, which requires $3rT_{\exp}$ as well as rT_{bp}.

Computing time of the four schemes in encryption and decryption phases is shown respectively in Figure 10 and Figure 11. It is easy to see that the computational time overhead of the proposed scheme is the smallest in these two stages. Because we introduce polynomials in the encryption and decryption algorithms, which are attribute independent and the time consumed does not increase with the number of attributes. Deng et al.[33] is the largest, because the bilinear pairing operation is used in encrypting each resource, which requires $(5r+3)T_{\exp}$. When decrypting resources, a large amount of T_{bp} is also required, and these bilinear pairing operations also include exponential operations, which increases the computation time significantly. A total of $(3r+1)T_{bp}$ and $rT_{\exp}$ are required. In the encryption phase, $(4r+1)T_{\exp}$ and $1T_{bp}$ are required by Xue et al.[34], while only $(n+2r+3)T_{\exp}$ is required by Li et al.[32], so the computational time overhead of Li et al.[32] is less than that of Xue et al.[34]. In the decryption phase, Xue et al.[34] is more advantageous than Li et al.[32] when the number of attributes is less than 12 because Li et al.[32] also needs more exp operations.

8 | CONCLUSIONS

In edge-cloud collaborative environment, a group key exchange and secure data sharing for federated learning is proposed. The scheme secures the transmission of model parameters between IoT terminals during the federated learning process. The model legitimizes public/private key of the terminal through key self-verification algorithm, which greatly guarantees the security of terminal key. For terminal identity leakage, an attribute-based cryptographic method is given for authentication, that is the terminal provides ciphertext attributes to the cloud server and obtains access rights corresponding to the attributes. The security of sharing the parameters of each model in the federated learning process is guaranteed. Terminal self-adaptive is achieved by defining access structures to meet shared resource access rights. In the condition of satisfying the access authority of model parameters, each terminal stores and downloads the shared ciphertext model parameters through the edge cloud server for federated learning. According to the experimental results, this scheme has lower computational complexity and less computation time. In the future, our team will continue to optimize this model and research deeply in the direction of ciphertext search.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China under Grant (No. 61971380), and the key technologies R&D Program of Henan Province (No. 232102211054, 232102211003, 222102210025), and the Key scientific research project

plans of higher education institutions in Henan Province (Nos. 23A520012, 22A520047, 21zx014), and Henan Postgraduate Joint Training Base Project (No. YJS2022JD08).

Financial disclosure

None reported.

Conflict of interest

The authors declare no potential conflict of interests.

References

1. Hazra A, Adhikari M, Amgoth T, Srirama SN. Intelligent service deployment policy for next-generation industrial edge networks. *IEEE Transactions on Network Science and Engineering* 2021; 9(5): 3057–3066.

2. Hazra A, Amgoth T. Ceco: Cost-efficient computation offloading of iot applications in green industrial fog networks. *IEEE Transactions on Industrial Informatics* 2021; 18(9): 6255–6263.

3. Su T, Shao S, Guo S, Lei M. Blockchain-based internet of vehicles privacy protection system. *Wireless Communications and Mobile Computing* 2020; 2020: 1–10.

4. Sun H, Tan Ya, Zhu L, Zhang Q, Li Y, Wu S. A fine-grained and traceable multidomain secure data-sharing model for intelligent terminals in edge-cloud collaboration scenarios. *International Journal of Intelligent Systems* 2022; 37(3): 2543–2566.

5. Zhang Y, Liang Y, Jia B, Wang P, Zhang X. A blockchain-enabled learning model based on distributed deep learning architecture. *International Journal of Intelligent Systems* 2022; 37(9): 6577–6604.

6. Pashamokhtari A. PhD forum abstract: Dynamic inference on IoT network traffic using programmable telemetry and machine learning. In: IEEE. ; 2020: 371–372.

7. Lo SK, Lu Q, Paik HY, Zhu L. FLRA: A reference architecture for federated learning systems. In: Springer. ; 2021: 83–98.

8. Ma J, Naas SA, Sigg S, Lyu X. Privacy-preserving federated learning based on multi-key homomorphic encryption. *International Journal of Intelligent Systems* 2022.

9. Lin Y, Zhang C. A Method for Protecting Private Data in IPFS. In: IEEE. ; 2021: 404–409.

10. Jia B, Zhang X, Liu J, Zhang Y, Huang K, Liang Y. Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT. *IEEE Transactions on Industrial Informatics* 2021; 18(6): 4049–4058.

11. Yang Y, Wei L, Wu J, Long C. Block-smpc: A blockchain-based secure multi-party computation for privacy-protected data sharing. In: ; 2020: 46–51.

12. Rao L, Xie Q, Zhao H. Data Sharing for Multiple Groups with Privacy Preservation in The Cloud. In: IEEE. ; 2020: 1–5.

13. Wang N, Cai Y, Fu J, Chen X. Information privacy protection based on verifiable (t, n)-Threshold multi-secret sharing scheme. *IEEE Access* 2020; 8: 20799–20804.

14. Piao C, Liu L, Shi Y, Jiang X, Song N. Clustering-based privacy preserving anonymity approach for table data sharing. *International Journal of System Assurance Engineering and Management* 2020; 11(4): 768–773.

15. Piao C, Hao Y, Yan J, Jiang X. Privacy protection in government data sharing: an improved LDP-based approach. *Service Oriented Computing and Applications* 2021; 15(4): 309–322.

16. Xuan S, Xiao H, Man D, Wang W, Yang W. A Cross-Domain Authentication Optimization Scheme between Heterogeneous IoT Applications.. *Wireless Communications & Mobile Computing* 2021.

17. Braeken A. Pairing free certified common asymmetric group key agreement protocol for data sharing among users with different access rights. *Wireless Personal Communications* 2021; 121(1): 307–318.

18. Zhao P, Huang Y, Gao J, Xing L, Wu H, Ma H. Federated Learning-Based Collaborative Authentication Protocol for Shared Data in Social IoV. *IEEE Sensors Journal* 2022; 22(7): 7385–7398.

19. Fan Q, Chen J, Deborah LJ, Luo M. A secure and efficient authentication and data sharing scheme for Internet of Things based on blockchain. *Journal of Systems Architecture* 2021; 117: 102112.

20. Singh C, Chauhan D, Deshmukh SA, Vishnu SS, Walia R. Medi-Block record: Secure data sharing using block chain technology. *Informatics in Medicine Unlocked* 2021; 24: 100624.

21. Jia X, Hu N, Yin S, Zhao Y, Zhang C, Cheng X. A2 chain: a blockchain-based decentralized authentication scheme for 5G-enabled IoT. *Mobile Information Systems* 2020; 2020.

22. Lv P, Wang Y, Wang Y, Liu C, Zhou Q, Xu Z. A highly reliable cross-domain identity authentication protocol based on blockchain in edge computing environment. In: IEEE. ; 2022: 1040–1046.

23. Zhan X, Cheng X, Guo W, Yin K, Lu X. An Distributed CA System: Identity authentication system in transnational railway transportation based on blockchain. In: IEEE. ; 2021: 989–994.

24. Wu S, Peng G, Gao Y, Chen J. An Efficient Anonymous Authentication Scheme for Medical Services Based on Blockchain. In: IEEE. ; 2021: 1–6.

25. Ahuja R, Mohanty SK. A scalable attribute-based access control scheme with flexible delegation cum sharing of access privileges for cloud storage. *IEEE Transactions on Cloud Computing* 2017; 8(1): 32–44.

26. Morales-Sandoval M, Cabello MH, Marin-Castro HM, Compean JLG. Attribute-based encryption approach for storage, sharing and retrieval of encrypted data in the cloud. *IEEE Access* 2020; 8: 170101–170116.

27. Kirupanithi DN, Antonidoss A. Efficient Data Sharing using Multi-authority Attribute Based Encryption in Blockchain. In: IEEE. ; 2021: 642–646.

28. Ezhil Arasi V, Indra Gandhi K, Kulothungan K. Auditable attribute-based data access control using blockchain in cloud storage. *The Journal of Supercomputing* 2022; 78(8): 10772–10798.

29. Ye Y, Zhang L, You W, Mu Y. Secure decentralized access control policy for data sharing in smart grid. In: IEEE. ; 2021: 1–6.

30. Ge C, Susilo W, Liu Z, Xia J, Szalachowski P, Fang L. Secure keyword search and data sharing mechanism for cloud computing. *IEEE Transactions on Dependable and Secure Computing* 2020; 18(6): 2787–2800.

31. Gafurov D, Hurum AE, Grovan MS. Access control tree for testing and learning. In: IEEE. ; 2021: 1106–1110.

32. Li B, Huang D, Wang Z, Zhu Y. Attribute-based access control for ICN naming scheme. *IEEE Transactions on Dependable and Secure Computing* 2016; 15(2): 194–206.

33. Deng H, Qin Z, Wu Q, Guan Z, Zhou Y. Flexible attribute-based proxy re-encryption for efficient data sharing. *Information Sciences* 2020; 511: 94–113.

34. Xue Y, Xue K, Gai N, Hong J, Wei DS, Hong P. An attribute-based controlled collaborative access control scheme for public cloud storage. *IEEE Transactions on Information Forensics and Security* 2019; 14(11): 2927–2942.