Wuhan University
Journal of Natural Sciences

2024, Vol.29 No.3, 228-238

Article ID 1007-1202(2024)03-0228-11 DOI https://doi.org/10.1051/wujns/2024293228

Cite this article: YANG Shuangxi. BDAE: A Blockchain-Based and Decentralized Attribute-Based Encryption Scheme for Secure Data Sharing[J]. *Wuhan Univ J of Nat Sci*, 2024, 29(3): 228-238.

BDAE: A Blockchain-Based and Decentralized Attribute-Based Encryption Scheme for Secure Data Sharing

YANG Shuangxi

School of Electronic Information, Wuhan University, Wuhan 430072, Hubei, China

Abstract: Ciphertext-policy attribute-based encryption (CP-ABE) is widely employed for secure data sharing and access control. However, its dependence on a single authority introduces security and performance challenges. Despite the existence of multi-authority CP-ABE approaches, persistent issues such as single points of failure and high computation cost on the user side remain. This study proposes a novel solution named blockchain-based and decentralized attribute-based encryption (BDAE) for data sharing. BDAE enhances traditional scheme by integrating blockchain and distributed key generation technology. The scheme employs an (n, t) threshold secret sharing algorithm, coupled with the Pedersen verifiable secret sharing method, for attribute key generation. This combination ensures key credibility, facilitates joint attribute management, and addresses single bottleneck and key verification issues. Integrated into a blockchain system, the scheme utilizes smart contracts for fine-grained access control and outsourced computing. Blockchain's decentralization and access logs make data sharing tamper-resistant and auditable. Moreover, simulation comparisons demonstrate that the scheme effectively reduces decryption overhead on the user side, meeting practical application requirements.

Key words: blockchain; attribute-based encryption; multi-authority; verifiable secret sharing; access control

CLC number: TP311.5

0 Introduction

With the continuous development of internet technology and technological innovation, cloud computing has revolutionized data sharing by providing extensive storage and management services to individuals and various industries. Despite its convenience, data sharing through cloud computing poses numerous security issues[1, 2]. One of the main issues is the lack of trust that users have in cloud service providers. These providers are vulnerable to external attacks from adversaries and internal threats posed by malicious employees, posing significant data security risks to data sharing. To address these challenges, cryptographic access control schemes such as Attribute-Based Encryption (ABE) and blockchain technology have emerged. Initially proposed by Bethencourt *et al*[3], the ciphertext-policy attribute-based encryption (CP-ABE) scheme maps user attributes and access policies into an access structure, allowing specific users who meet the access structure criteria to decrypt data. This feature aligns well with the require-

Received date: 2024-01-24

Biography: YANG Shuangxi, male, Master candidate, research direction: blockchain technology. E-mail: 2020202120070@whu.edu.cn

ments of data access control in cloud environments. Subsequently, researchers have proposed various data sharing solutions by applying this algorithm in different domains. However, many existing schemes[4-6] suffer from the issue of excessive authority and a single point of bottleneck, where all parties are compelled to trust a single attribute authority. This vulnerability can lead to single points of failure and performance bottlenecks. In practical applications, attributes are typically managed in a distributed manner by different institutions. To solve this issue, Chase *et al*[7,8] introduced the first multi-authority ABE scheme. This scheme manages multiple unrelated attribute domains through different authorities, facilitating the distribution of user attributes across multiple management institutions. Subsequently, several extension schemes[9-12] were proposed. However, they still face the challenge of a single point of bottleneck. If one institution goes down or fails, all attributes managed by that institution become invalid. Li *et al*[13] proposed a TMACS (threshold multi-authority access control) scheme that leverages threshold secret sharing algorithms to achieve joint attribute management, thus addressing the single point of bottleneck issue and enhancing system robustness. However, this scheme requires users to communicate with multiple institutions, which increases the communication costs.

Qin *et al*[14] suggested using blockchain and smart contracts to connect users and attribute authorities (AAs), reducing user-side computational costs. However, key verification is lacking during attribute key generation, compromising key security and reliability.

To address the challenges associated with cross-domain collaboration among multiple authorities and mitigate single points of failure, this paper introduces BDAE—a blockchain-based and decentralized attribute-based encryption access control scheme. By utilizing smart contracts, BDAE establishes mutual trust among various authority institutions. Collaborative computation of attribute sub-tokens facilitates the generation of decryption tokens for users, enabling decentralized data access control. Additionally, blockchain technology ensures trustworthy and tamper-resistant access logs, allowing data owners to monitor user access behavior seamlessly. In summary, our contributions can be summarized as follows:

1) Proposal of BDAE, a decentralized access control scheme based on Hyperledger Fabric[15] and distributed key generation. This approach ensures trusted key management, addressing the single point of bottleneck and key verification issues.

2) Utilization of smart contracts and blockchain's identity management features to enable the distributed issuance of attribute certificates for users, providing fine-grained data access control and sharing solutions.

3) Exploitation of the decentralized and auditable nature of blockchain technology, ensuring tamper-resistant data sharing processes and enhancing overall security and reliability.

4) Comprehensive analysis of the system's functionalities and performance, demonstrating that the scheme reduces decryption overhead on the user side without compromising privacy, thereby meeting the requirements of practical applications.

1 Overview

1.1 System Model

As illustrated in Fig. 1, the system comprises five distinct entities and a blockchain network, including multiple Certificate Authorities (CAs), multiple AAs, Data Owner (DO), Data User (DU), the Interplanetary File System (IPFS) for data storage, and a peer-to-peer blockchain network.

CAs: The blockchain network adopts a distributed architecture with multiple organizations, each equipped with a CA. The CA is responsible for issuing attribute certificates to members within the organization, such as users and AAs, and generating network identity credentials for them. It is important to note that the CA does not participate in the generation of attribute keys for users and AAs.

AAs: They are responsible for attribute management and attribute key generation. Users can request attribute tokens from AAs based on the attributes carried by their certificates. In contrast to traditional multi-authority CP-ABE systems, we optimized the expression strategy for attributes and each attribute name possesses multiple attribute values, which are jointly managed by multiple authorities. This creates a many-to-many mapping relationship between attribute sets and authorities. The attribute management model is illustrated as Fig. 2.

IPFS: IPFS[16] is a content-addressable and peer-to-peer hypermedia distribution protocol that aims to create a persistent and distributed storage and file sharing network. Nodes in the IPFS network collectively form a distributed file system. IPFS is a replacement for traditional

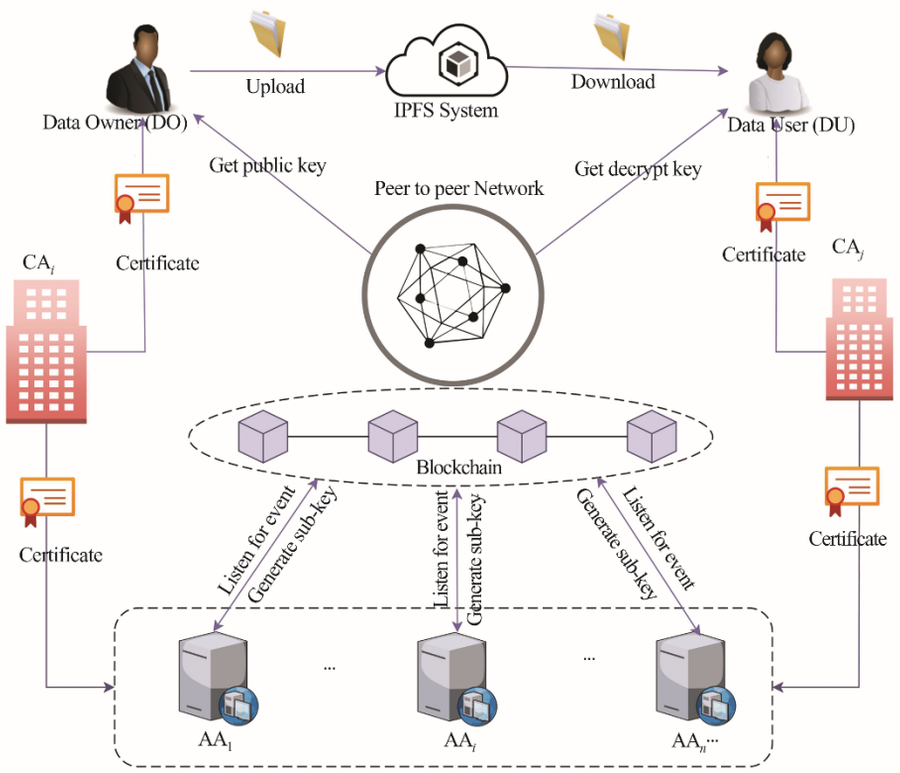

Fig. 1 Blockchain-based multi-authority access control system model

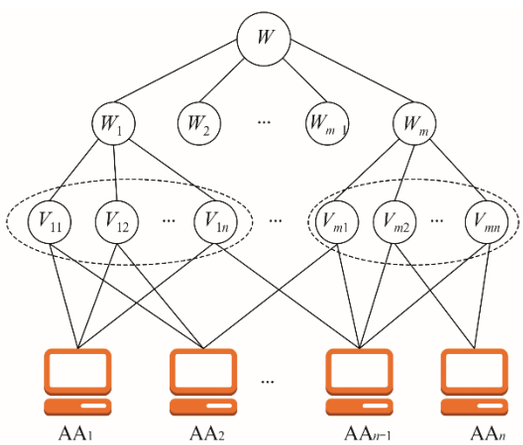

Fig. 2 Attribute management model

cloud service providers and provides data storage services for data owners in the system, and users can retrieve files from the network using only the Content Identifier (CID).

DO: To enhance data encryption efficiency, all DOs should encrypt the original data using symmetric encryption algorithms such as AES, DES, etc., before uploading files to IPFS. Subsequently, the data access policy is defined, and the symmetric encryption key is encrypted using attribute-based encryption algorithms. Finally, CID and the encrypted key are uploaded to the blockchain.

DU: By applying for identity credentials from the CA, each user obtains a certificate with globally unique ID, referred to as globally unique identity (GID). All users can download ciphertext from IPFS freely, but only those who meet the access policy can obtain decryption tokens through attributes in certificates, thereby completing data decryption.

Blockchain: The blockchain provides a decentralized, tamper-resistant, and cross-domain collaborative data interaction platform. Communication between all entities is carried out through the blockchain network. The blockchain ledger stores the publicly initialized parameters and metadata, ensuring data audibility and reliability.

1.2 Workflow

The workflow of the system comprises five different stages: Certificate Generation, System Initialization, Encrypt, Token Generation, and Decrypt. As illustrated in Fig. 3, each stage is represented by a different color.

① **Certificate Generation**: The system employs collaborative network topology across various organizations and domains. Fabric CA issues certificates to members of each organization. Administrators within each organization submit registration requests, and Fabric CA

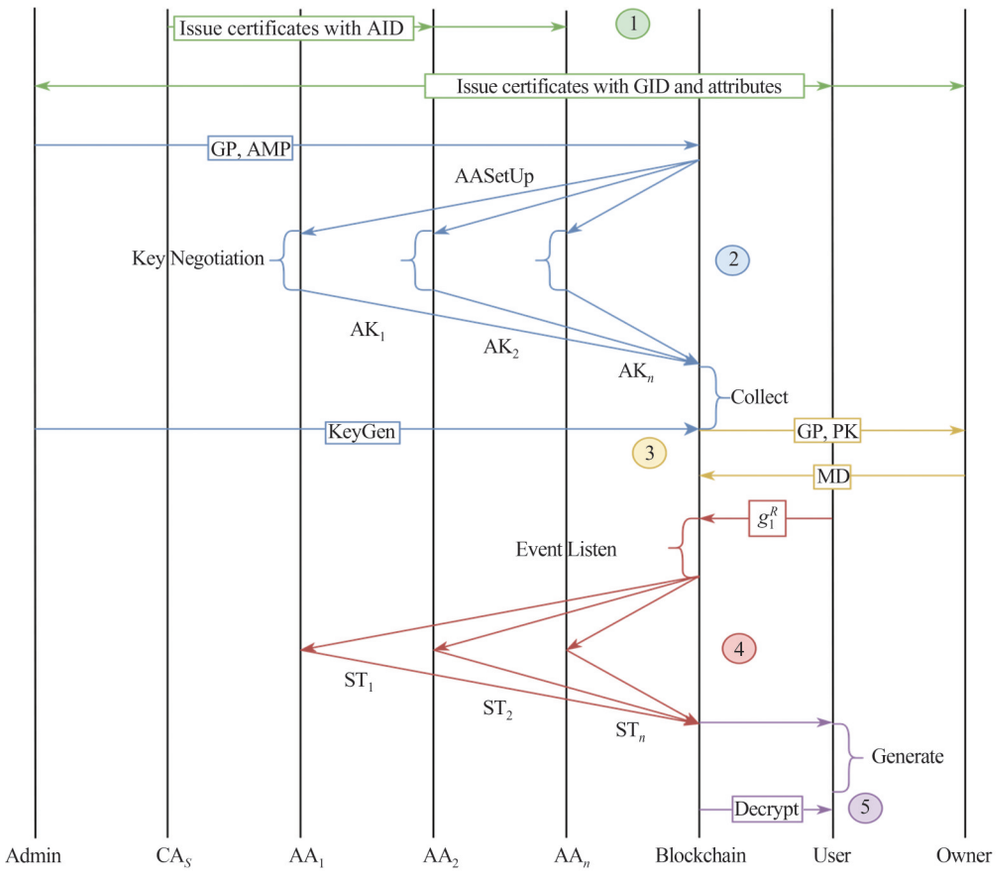

Fig. 3 Procedure flow of the system mode

processes these requests to generate certificates with a GID and associated attributes. These certificates serve as the sole access tokens for members to engage with the blockchain network. Fabric CA can also invalidate certificates upon request or in case of user violations.

② **System Initialization**: As depicted below, system initialization consists of three steps.

1) GlobalSetUp(λ)→GP

λ is a safe parameter for determining the size of a finite group. The system administrator runs initialization algorithm to generate the global parameter GP. Subsequently, customized attribute management policies for authorities are formulated and then uploaded to the blockchain. In the real life, an attribute usually has multiple attribute values. For example, a person's work department can be production, sales, or logistics. In order to facilitate the management of attributes, this paper designs a convenient data structure to store policies. These policies follow the format depicted in Fig. 4, for example, the attribute name W_i has k attribute values, which are jointly managed by n_i AAs, and the key reconstruction threshold is t_i.

attribute name	attribute value	authority list	threshold
W_i	$\{V_{i1},V_{i2},\cdots,V_{ik}\}$	$\{AA_1,AA_2,\cdots,AA_{n_i}\}$	t_i

Fig. 4 An instance of attribute management policy

2) AASetUp (GP, AID)→(AA)

Each AA retrieves the GP from the blockchain and employs it alongside its authority GID to execute the algorithm, thus formally initializing itself as an authority entity.

3) KeyNegotiate (AMP)→({subPK},{subSK})

All AAs, upon acquiring the all AMPs from the blockchain, conduct key negotiation based on the policies. The negotiated keys serve as private keys for attributes under their management domain, and the corresponding sub-public keys are uploaded to the blockchain.

4) KeyGenerate({subPK})→(GPK)

The smart contract is responsible for gathering sub-public keys from various authorities on-chain. Subsequently, the system administrator merges the set of sub-public keys {subSK} off-chain to generate the global public key GPK, which is then uploaded to the blockchain.

③ **Encryption**: After retrieving GP and GPK from the blockchain, the data owner encrypts his data and uploads the ciphertext to IPFS. The metadata MD is also uploaded to the blockchain and stored in JSON format as shown in Fig. 5. The index of the ciphertext in IPFS is represented by CID_{IPFS}, serving as the sole proof of the ciphertext's retrieval.

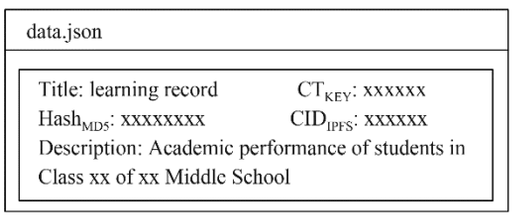

Fig. 5 An instance of meta data

1) SymEncryption(M)→CT

Due to the complexity of operations like pairings in ABE, efficiency is lower compared to symmetric encryption. To ensure efficient data encryption, a hybrid encryption approach is adopted, and the ciphertext CT is uploaded to IPFS.

2) Encrypt (K, ($\boldsymbol{A}$, ρ), GP, GPK)→CT_{KEY}

Data owner executes this algorithm, inputting the symmetric encryption key K, access structure ($\boldsymbol{A}$, ρ), global parameter GP, and global public key GPK to generate the key ciphertext CT_{KEY}.

④ **Token Generation**: To decrypt data, user accesses the blockchain network for a decryption token. The smart contract identifies user attributes from his certificate, triggering relevant events and sending token generation requests to respective authorities according to the attribute management policy. Multiple authorities monitor and respond by sending sub-tokens to the user.

SubTokenGen(g_1^R, subSK, GP, GID)→(subTK)

The algorithm is executed off-chain by multiple AAs. Differing from traditional multi-authority CP-ABE schemes where attribute sub-keys are directly distributed to users by authorities, our proposed method transmits sub-keys to users through the blockchain network. Users are required to provide the blinding factor g_1^R, $R \in \mathbb{Z}_p$ to prevent attribute sub-keys leakage. Once obtaining the user's GID, the authority uses its subSK to generate an attribute sub-token subTK which is then transmitted to the blockchain.

⑤ **Decryption**: To generate an attribute token, the data user needs to merge sub-tokens for the same attribute name retrieved them from the blockchain. Then user can decrypt the key ciphertext CT_{KEY} and ultimately decrypt the ciphertext CT obtained from IPFS using a decryption algorithm.

1) TokenGen({subTK})→(AT)

This algorithm is executed by users to merge sub-token set for the same attribute, generating an attribute token AT corresponding to the set.

2) Decrypt({AT}, CT_{KEY}, R, GP)→(KEY)

The algorithm takes the token set {AT} corresponding to the data user's attribute set, the key ciphertext CT_{KEY}, and the global parameter GP as input. It then derives the symmetric key KEY used for decrypting the original data's ciphertext.

2 Data Sharing Scheme

We introduce a blockchain-based scheme designed to facilitate communication among multiple AAs, streamlining cross-domain attribute management through the use of smart contracts. Users exclusively interact with the blockchain, simplifying communication complexities. On the blockchain, three distinct smart contracts operate in separate channels—Attribute Management Contract (AMC), Key Management Contract (KMC), and Metadata Management Contract (DMC)—ensuring data isolation and enhancing security. Off the blockchain, the key negotiation function addresses the single-point bottlenecks found in traditional schemes. Below, we present a detailed data access control scheme.

2.1 System Initialization

GlobalSetUp: The administrator selects cyclic groups G_1, G_2, and G_T of order p. A bilinear mapping e: $G_1 \times G_2 \rightarrow G_T$ and a Hash function: H $\{0,1\}^* \rightarrow G$ are defined. The Hash function maps GID to G elements. The resulting global parameters, GP= (p, g_1, g_2, g_T, H), are established. Then the system administrator defines attribute management policies, uploading them to the blockchain using smart contract AMC.

AASetUp: Upon obtaining the global parameters GP from the blockchain, each attribute authority AA initializes an attribute authority instance using its own GID. This step prepares for generating attribute key SK in subsequent stages.

KeyNegotiate: During initialization, AAs get attribute management policies from the blockchain and negotiate keys off-chain based on the policy. We upgraded the traditional secret sharing scheme[17] with a Pedersen-based verifiable method[18], significantly boosting secu-

rity compared to previous approaches[13,14]. This ensures both decentralization and verification, preventing any single authority from possessing the master keys and avoiding the scenario where other authorities receive fragments of incorrect keys. Importantly, in achieving information-theoretic security, the scheme has an information rate of 1, surpassing Pedersen's 1/2. The process involves four stages:

1) Sub-share generation

For each attribute value, key negotiation is conducted among cooperating AAs. Assuming attribute value V_{i1} under attribute W_i is managed by $\{AA_1, AA_2, \cdots, AA_{n_i}\}$, each AA_k ,$k \in \{1,2,\cdots,n_i\}$selects two random numbers α_i , $\beta_i \in \mathbb{Z}_p$ as their sub-secrets. Then they respectively generate two polynomials $f_k(x)$, $f_k'(x)$ of degree t_i-1 with coefficients randomly chosen from $\mathbb{Z}_p$, satisfying $f_k(0)=\alpha_i$ and $f_k'(0)=\beta_i$. Thus, the master private key for V_{i1} can be computed as follow:

$$\alpha_{i1}=\sum_{i=1}^{n_i}\alpha_i, \beta_{i1}=\sum_{i=1}^{n_i}\beta_i \tag{1}$$

However, AAs cannot directly access each other's sub-secrets, so AA_k needs to compute sub-shares $S_{kj}=f_k(x_j)$, $S_{kj}'=f_k'(x_j)$, $j \in \{1,\cdots,n_i\}\backslash\{k\}$ for others. Simultaneously, it computes its own sub-shares $S_{kk}=f_k(x_k)$, $S_{kk}'=f_k'(x_k)$, where $x \in \mathbb{Z}_p$ represents the identifier of each AA during key negotiation.

2) Sub-share exchange

After generating sub-shares, AAs proceed to exchange them. Prior research[13,14,17] did not focus on verifying the correctness of share exchange, neglecting potential tampering or malicious alterations during distribution. To tackle this issue, we implement the Verifiable Secret Sharing (VSS) scheme.

● Public commitment

Given g and h as two generators of the group G_q, where $\log_g^h$ is unknown and q represents the order of elliptic curve in user's public keys. Let $E(s,t)=g^s h^t$ be the commitment. So AA_k is required to compute the commitment C and reveal it to the other AAs.

$$C=\{E(f_{k0},f_{k0}'), E(f_{k1},f_{k1}'), \cdots, E(f_{k(t_i-1)},f_{k(t_i-1)}')\} \tag{2}$$

● Distribute sub-share

The AA_k encapsulates sub-shares within a structure named "deal", expressed as deal = $\{C, f_k(x_j), f_k'(x_j), g, h\}$, and encrypts it using the AES-GCM symmetric encryption algorithm. During encryption, the authority generates a signature for identity verification using its private key from blockchain. The sealing process is detailed in Algorithm 1. AES-GCM mode, chosen for its support of parallel encryption and decryption alongside identity verification, is preferred over other AES encryption modes.

Algorithm 1: Seal the deal

Input: deal, SK_{AA_k}, PK_{AA_j}
Output: seal, Error
dhSec ← Pick(RandomStream());
dhPub ← PKGen(dhSec, g);
signature, err ← Sign(dhPub, SK_{AA_k});
if err ≠ nil **then**
  return nil, err
end
// Diffie-Hellman key exchange protocol
dhKey ← dhKeyGen(dhSec, PK_{AA_j});
nonce ← io.ReadFull();
encDeal, err ← AESencrypt(dhKey, deal);
if err ≠ nil **then**
  return nil, err
end
seal={encDeal, dhPub, signature, nonce};
return seal

● Verify sub-share

The consortium of AA_j ($j \in \{1,\cdots,n_i\}\backslash\{k\}$) who manage the attribute value V_{i1} need to perform the following steps after receiving the seal from AA_k. First, AA_j validates the signature of the temporary public key dhPub using the AA_k's public key. Upon successful verification, AA_j reconstructs the temporary encryption key dhKey using its own private key and decrypts the ciphertext to obtain the deal. The process is demonstrated in Algorithm 2.

Algorithm 2: Unseal the deal

Input: seal, PK_{AA_k}, SK_{AA_j} ,
Output: deal, Error
err ← Verify(dhPublic, PK_{AA_k}, signature);
if err ≠ nil **then**
  return nil, error.New("fail to verify signature");
end
// Diffie-Hellman key exchange protocol
dhKey ← dhKeyGen(dhPublic, SK_{AA_j});
deal ← AESDecrypt(dhKey, encDeal, nonce);
if err ≠ nil **then**
  return nil, error.New("fail to decrypt encDeal")
end
return deal

If the decryption of the ciphertext is successful, AA_j employs the commitment C to validate the correctness of the sub-share. The validation equation is expressed as follows:

$$\begin{aligned} &E(f_k(x_j),f'_k(x_j))=g^{f_k(x_j)}h^{f'_k(x_j)}\\ &=E(f_{k0},f'_{k0})\cdot E(f_{k1},f'_{k1})^{x_j}\cdots E(f_{k(t-1)},f'_{k(t-1)})^{x_j^{t_i-1}}\\ &=\prod_{n=0}^{t_i-1}E_n^{x_j^n} \end{aligned} \tag{3}$$

According to Ref. [18], we can calculate the information rate:

$$\frac{\text{size of secret}}{\text{size of share}}=\frac{2}{2}=1 \tag{4}$$

3) Master share generation

After receiving n_i-1 sub-shares (S_{jk}, S_{jk}') from other n_i-1 AAs, AA_k computes its master share as the attribute private key share for V_{i1}, which can be computed as follows:

$$\begin{cases} \mathrm{SK}_{i1,k}=\sum_{j=1}^{n_i}S_{jk}\\ \mathrm{TK}_{i1,k}=\sum_{j=1}^{n_i}S'_{jk} \end{cases} \tag{5}$$

The attribute public key share is computed as:

$$\begin{cases} \mathrm{SP}_{i1,k}=e(g_1,g_2)^{\mathrm{SK}_{i1,k}}\\ \mathrm{TP}_{i1,k}=g_2^{\mathrm{TK}_{i1,k}} \end{cases} \tag{6}$$

Next, AA_k securely saves the private key share while publicly uploading the public key share to the blockchain.

4) Global pub-key generation

Once the blockchain receives all attribute public key shares, the system administrator utilizes the smart contract KMC to gather t_i shares, generating global pub-key generation GPK. And then it will be uploaded to the blockchain for use by the data owner. The public key generation algorithm is a variant of Lagrange interpolation. For example, the public key $\mathrm{SP}_{V_{i1}}$ for attribute value V_{i1} is computed as follows:

$$\begin{aligned} \mathrm{SP}_{i1}&=\prod_{k=1}^{t_i}\mathrm{SP}_{i1,k}^{\prod_{j\neq k}^{t_i}\frac{x_j}{x_j-x_k}}\\ &=\prod_{k=1}^{t_i}e(g_1,g_2)^{(\mathrm{SK}_{i1,k}\prod_{j=1,j\neq k}^{t_i}\frac{x_j}{x_j-x_k})}\\ &=e(g_1,g_2)^{\sum_{k=1}^{t_i}(\mathrm{SK}_{i1,k}\prod_{j=1,j\neq k}^{t_i}\frac{x_j}{x_j-x_k})}\\ &=e(g_1,g_2)^{\alpha_{i1}} \end{aligned} \tag{7}$$

and

$$\begin{aligned} \mathrm{TP}_{i1}&=\prod_{k=1}^{t_i}\mathrm{TP}_{i1,k}^{\prod_{j\neq k}^{t_i}\frac{x_j}{x_j-x_k}}\\ &=\prod_{k=1}^{t_i}e(g_1,g_2)^{(\mathrm{TK}_{i1,i}\prod_{j=1,j\neq k}^{t_i}\frac{x_j}{x_j-x_k})}\\ &=e(g_1,g_2)^{\sum_{k=1}^{t_i}(\mathrm{TK}_{i1,k}\prod_{j=1,j\neq k}^{t_i}\frac{x_j}{x_j-x_k})}\\ &=e(g_1,g_2)^{\beta_{i1}} \end{aligned} \tag{8}$$

And the calculation for TP_{i1} is the same as that for SP_{i1}. Thus, the GPK is a collection of all attribute public keys in the system, denoted as GPK= {SP, TP}.

2.2 Encrypt

After obtaining the GP and GPK, the data owner defines an access policy and employs the attribute-based encryption algorithm to encrypt the symmetric encryption key K. An access structure can be represented as an LSSS access structure ($\boldsymbol{A}$, ρ), where $\boldsymbol{A}$ represents an $m\times l$ matrix. The ρ maps the x-th row of M to an attribute $\rho(x)\in A_{\mathrm{tt}1},\cdots,A_{\mathrm{tt}U}$. For each row of $\boldsymbol{A}$, the data owner selects a random vector $\boldsymbol{v}=(s, y_2, y_3,\cdots, y_l)\in\mathbb{Z}_p$, where s is a secret value and $y_2, y_3, \cdots, y_l$ are used to share the secret s. Let $\lambda_x=A_x\boldsymbol{v}$, where A_x represents the row x of $\boldsymbol{A}$. The data owner selects a random vector $\boldsymbol{w}=(0, w_1, w_2, \cdots, w_l)\in\mathbb{Z}_p$, and computes $w_x=A_x\boldsymbol{w}$. Random values $r_1, r_2, \cdots, r_m\in\mathbb{Z}_p$ are chosen. The ciphertext is then calculated as follows:

$$\begin{aligned} &C_0=Ke(g_1,g_2)^S,\ \ C_{1,x}=e(g_1,g_2)^{\lambda_x}e(g_1,g_2)^{\alpha_{\rho(x)}r_x},\\ &C_{2,x}=g_2^{r_x},\ \ C_{3,x}=g_2^{\beta_{\rho(x)}r_x}g_2^{w_x}\forall x \end{aligned} \tag{9}$$

2.3 Token Generation

Before decrypting data, users must acquire decryption sub-tokens. Our scheme optimizes communication efficiency by allowing users to interact exclusively with the blockchain network. The smart contract KMC automates the extraction of attribute values from the user's certificate and facilitates communication with multiple AAs to obtain sub-tokens.

SubTokenGen: Each AA_k sets up an event listener off-chain. When a user triggers the token generation function through smart contract KMC, an immediate event is activated. For example, if the attribute value V_{i1} embedded in the user's certificate corresponds to AA_k's attribute management domain, AA_k generates the relevant sub-token. The sub-token can be expressed as:

$$\mathrm{ST}_{i1,k}=(g_1^R)^{\mathrm{SK}_{i1,k}}H(\mathrm{GID})^{\mathrm{TK}_{i1,k}} \tag{10}$$

where $\mathrm{SK}_{i1,k}$, $\mathrm{TK}_{i1,k}$ are the attribute sub-private keys possessed by AA_k. Subsequently, AA_k sends the sub-token to the blockchain using the smart contract.

TokenGen: Since each attribute value corresponds to multiple sub-tokens, data user needs to aggregate sub-tokens belonging to the same attribute value. The aggregation algorithm is a variant of Lagrange interpolation, as shown for attribute value V_{i1}:

$$\mathrm{AT}_{i1,\mathrm{GID}}=\prod_{k=1}^{t_i}\mathrm{ST}_{i1,k}^{\prod_{j=1,j\neq k}^{t_i}\frac{x_j}{x_j-x_k}}=(g_1^R)^{\alpha_{i1}}H(\mathrm{GID})^{\beta_{i1}} \tag{11}$$

The user's tokens for all attribute values can be expressed as a token collection $\{\mathrm{AT}_{\mathrm{GID}}\}$.

2.4 Decrypt

The decryption algorithm can be executed by data users who meet the access policy. The user U who has obtained metadata MD and global parameters GP from the blockchain must use the token set $\mathrm{AT}_{\mathrm{GID}}$ and blinding factor R as input. Let $\boldsymbol{A}_U$ be a sub-matrix of $\boldsymbol{A}$, where $\boldsymbol{A}_U$ is an $n\times l$ matrix. Each row of this matrix corresponds to an attribute in U's attribute set S_U. If S_U satisfies the access structure $(\boldsymbol{A}, \rho)$, then there always exists a suitable vector $\boldsymbol{h}=(h_1,h_2,\cdots, h_l)\in\mathbb{Z}_p^l$ such that $\vec{\boldsymbol{e}}=A_U\cdot h$, where $\vec{\boldsymbol{e}}=(1,0,\cdots,0)\in\mathbb{Z}_p^l$.

To alleviate the computational burden on users, it is essential for the blockchain to shoulder a portion of the associated computational costs. Utilizing the token set $\mathrm{AT}_{\rho(x),\mathrm{GID}}$ of user U, for each such x, user invoke smart contract to compute:

$$P=\prod_x C_{1,x}^{h_x},\ Q=\prod_x\left(\frac{e(H(\mathrm{GID}),C_{3,x})}{e(\mathrm{AT}_{\rho(x),\mathrm{GID}},C_{2,x})}\right)^{h_x} \tag{12}$$

The blockchain returns the calculation result (P, Q) to user. According to it, users only need to perform calculations based on the following formula:

$$\begin{aligned}&\prod_x\left(\frac{C_{1,x}^R\cdot e(H(\mathrm{GID}),C_{3,x})}{e(\mathrm{AT}_{\rho(x),\mathrm{GID}},C_{2,x})}\right)^{h_x}\\&=\prod_x\left(e(g_1,g_2)^{\lambda_x R}e\left(H(\mathrm{GID}),g_2\right)^{w_x}\right)^{h_x}=P^RQ\end{aligned} \tag{13}$$

Then, using the vector $\boldsymbol{h}$, we can compute equation:

$$\begin{aligned}\boldsymbol{s}&=(1,0,\cdots,0)^{\mathrm{T}}(s,y_2,\cdots,y_l)\\&=\vec{\boldsymbol{e}}^{\mathrm{T}}\cdot\vec{\boldsymbol{v}}=\boldsymbol{h}^{\mathrm{T}}\cdot\boldsymbol{A}_U^{\mathrm{T}}\cdot\vec{\boldsymbol{v}}\\&=\boldsymbol{h}^{\mathrm{T}}\cdot\vec{\boldsymbol{\lambda}}_U\\&=\sum_{x\in U}h_x\cdot\lambda_x\end{aligned} \tag{14}$$

and

$$\begin{aligned}&\sum_{x\in U}w_x\cdot h_x=\boldsymbol{w}^{\mathrm{T}}\cdot\boldsymbol{A}_U\cdot\boldsymbol{h}\\&=(0,w_1,\cdots,w_l)^{\mathrm{T}}\cdot(1,0,\cdots,0)\\&=0\end{aligned} \tag{15}$$

Therefore, we can compute the equation:

$$P^RQ=e(g_1,g_2)^{RS} \tag{16}$$

Finally, we obtain the decrypted message:

$$K=\frac{C_0}{(e(g_1,g_2)^{RS})^{1/R}}=\frac{C_0}{e(g_1,g_2)^S} \tag{17}$$

3 Scheme Analysis

3.1 Security and Robustness

Each attribute token in the system is jointly managed by n AAs. Due to the t threshold parameter, adversaries require compromising t or more AAs to acquire decryption keys for a specific attribute. Similarly, the system's normal operation necessitates t or more AAs to be online. This section analyzes the system's security and robustness.

Security: The system's security depends on the management of attribute values by AAs. The security assessment focuses on the probability, denoted as λ, representing an adversary's likelihood of illicitly obtaining an attribute key by compromising a specific AA. This metric forms the basis of the system's security analysis

$$\sum_{i=0}^{t-1}\binom{i}{n}\lambda^i(1-\lambda)^{n-i} \tag{18}$$

Robustness: System resilience is measured by the probability of continuous operation despite AA failures, Instances like crashes or offline status in AAs could obstruct users' access to managed sub-keys. Let μ denote the likelihood of individual AA operational failures. The calculation for system robustness is as follows:

$$\sum_{i=t}^{n}\binom{i}{n}\mu^i(1-\mu)^{n-i} \tag{19}$$

To understand security and robustness better, we varied parameter values in comparative experiments (Fig. 6). For a fixed corruption probability λ, increasing t enhances system security, yet larger t values do not always ensure better security. On the other hand, higher n, the number of AAs, improves system robustness even with higher corruption probabilities. However, as n increases, system security decreases. Thus, balancing n and t is crucial for optimal performance in practical applications.

3.2 Comparison of Functionality

In Table 1, we compare the scheme with previous multi-authority ABE schemes. The evaluation metrics include bilinear groups, auditability, access structure, robustness, identity management, and key verification. Table 1 shows that the proposed scheme offers more comprehensive functionality compared with previous schemes.

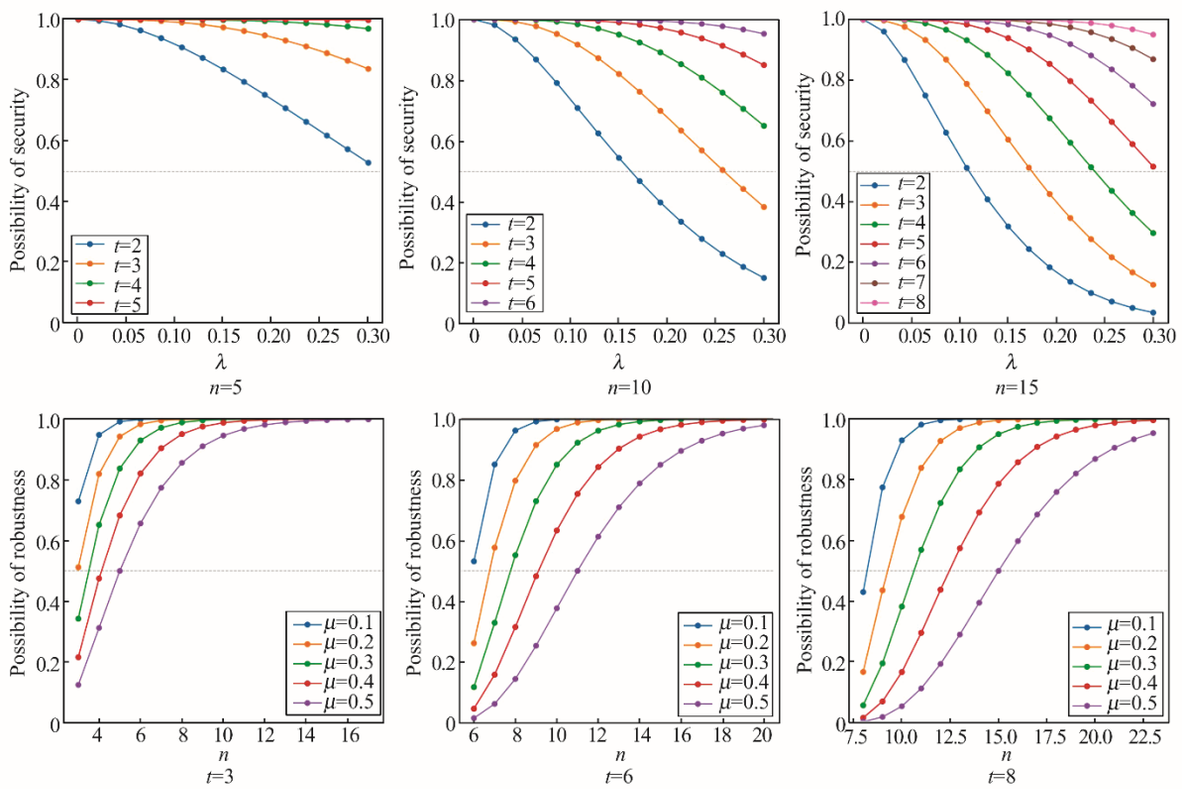

Fig. 6 Probability of security and robustness

Table 1 Comparison of various multi-authority ABE schemes

Scheme	Auditability	Access structure	Robustness	Identity management	Key verification
TMACS[13]	No	LSSS	Yes	No	No
LW[9]	No	LSSS	No	No	No
DIPPE[12]	No	Threshold	No	No	No
Proposed	Yes	LSSS	Yes	Yes	Yes

3.3 Performance Analysis

We performed simulations to assess the computational costs of our scheme in comparison to previous ones. To ensure fair experimental comparisons, all schemes were implemented using 256-bit BN elliptic curves with prime order. The experimental setup utilized a physical machine with an AMD Ryzen7 4800U CPU. All tests were conducted on a virtual machine running CentOS 7.9, with an allocation of 8GB of memory. Additionally, all algorithms were implemented using the Golang programming language.

Given diverse access policies, we tested computational overhead under two extreme scenarios. To minimize performance impact, we varied attributes per test round, repeating 100 times at different intervals for averages. As shown in Fig. 7, the results show that there is a linear rise in encryption time with increasing attributes for all schemes. Although our scheme exhibits a slight performance advantage over LW[9], it notably surpasses DIPPE[12].

In decryption, our proposed scheme notably reduces user-side computational costs. In fact, due to the blinding factor, even if partial decryption calculations are handed over to smart contracts or third-party servers, no information will be leaked, making the decryption process both secure and efficient. By seamlessly performing outsourced calculations without compromising privacy, users only need to execute an exponential operation and a simple multiplication for final decryption. Importantly, this execution time remains constant, independent of attribute count, ensuring efficient decryption on the user side.

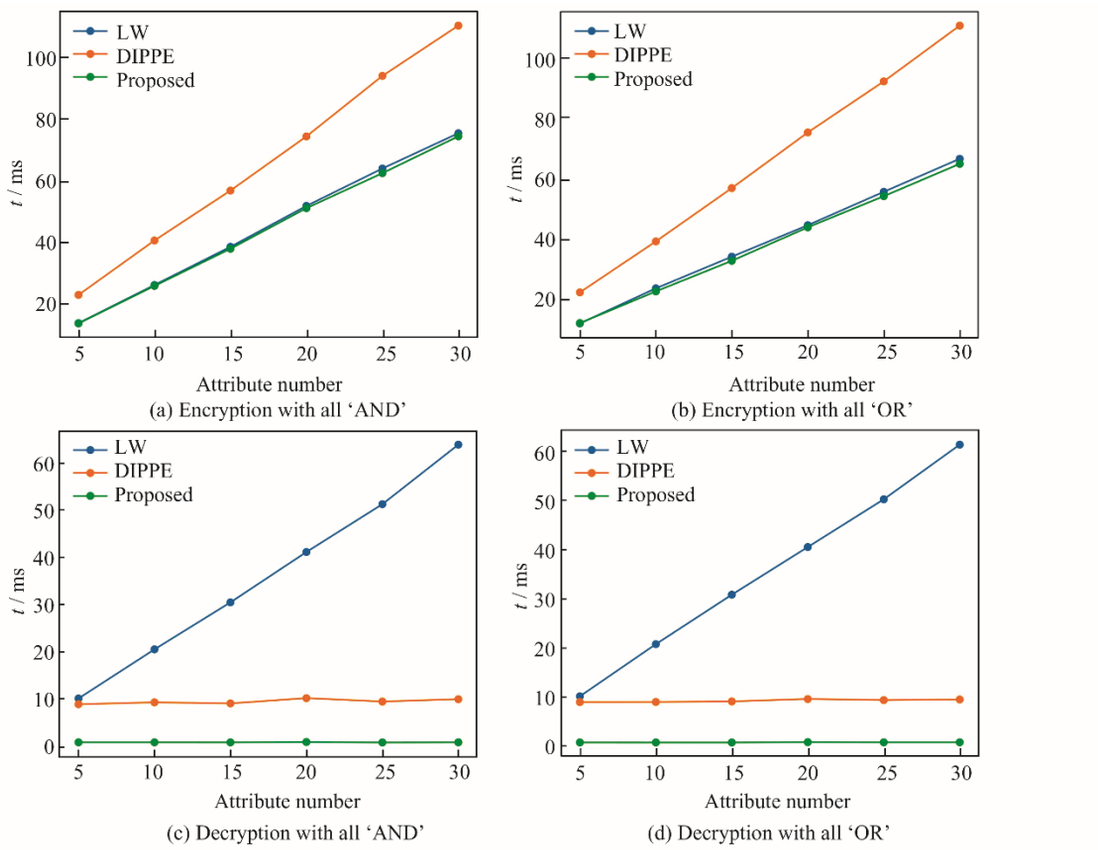

Fig. 7 Comparison of user side computational overhead under different access policies

4 Conclusion

In this paper, we propose a multi-authority attribute-based encryption scheme based on blockchain and distributed key generation which is called BDAE for data access control. Our work enhances system robustness by managing each attribute with multiple organizations and solving the single bottleneck problem. The use of blockchain technology ensures the security and traceability of the data sharing process. For future work, we will focus more on reducing user overhead in encryption calculation.

References

[1] Yang P, Xiong N X, Ren J L. Data security and privacy protection for cloud storage: A survey[J]. *IEEE Access*, 2020, **8**: 131723-131740.

[2] Akhtar D N, Kerim D B, Perwej D Y, *et al*. A comprehensive overview of privacy and data security for cloud storage[J]. *International Journal of Scientific Research in Science, Engineering and Technology*, 2021, **8**(5): 113-152.

[3] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption[C]//2007 *IEEE Symposium on Security and Privacy* (*SP '07*). New York: IEEE, 2007: 321-334.

[4] Xue Y J, Xue K P, Gai N, *et al*. An attribute-based controlled collaborative access control scheme for public cloud storage [J]. *IEEE Transactions on Information Forensics and Security*, 2019, **14**(11): 2927-2942.

[5] Venema M, Alpár G. TinyABE: Unrestricted ciphertext-policy attribute-based encryption for embedded devices and low-quality networks[C]//*International Conference on Cryptology in Africa*. Berlin, Heidelberg: Springer-Verlag, 2022: 103-129.

[6] Yang X D, Li T, Pei X Z, *et al*. Medical data sharing scheme based on attribute cryptosystem and blockchain technology [J]. *IEEE Access*, 2020, **8**: 45468-45476.

[7] Chase M. Multi-authority attribute-based encryption[C]// *Theory of Cryptography Conference*. Berlin, Heidelberg: Springer-Verlag, 2007: 515-534.

[8] Chase M, Chow S S M. Improving privacy and security in multi-authority attribute-based encryption[C]//*Proceedings of the* 16*th ACM Conference on Computer and Communications Security*. New York: ACM, 2009: 121-130.

[9] Lewko A, Waters B. Decentralizing attribute-based encryption[C]//*Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Berlin, Heidelberg: Springer-Verlag, 2011: 568-588.

[10] Zhang L Y, Ren J, Mu Y, *et al*. Privacy-preserving multi-authority attribute-based data sharing framework for smart grid[J]. *IEEE Access*, 2020, **8**: 23294-23307.

[11] Tao J Y, Ling L. Practical medical files sharing scheme based on blockchain and decentralized attribute-based encryption[J]. *IEEE Access*, 1917, **9**: 118771-118781.

[12] Michalevsky Y, Joye M. Decentralized policy-hiding ABE with receiver privacy[C]//*European Symposium on Research in Computer Security*. Berlin, Heidelberg: Springer-Verlag, 2018: 548-567.

[13] Li W, Xue K P, Xue Y J, *et al*. TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2016, **27**(5): 1484-1496.

[14] Qin X M, Huang Y F, Yang Z, *et al*. A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing[J]. *Journal of Systems Architecture*, 2021, **112**: 101854.

[15] Androulaki E, Barger A, Bortnikov V, *et al*. Hyperledger fabric: A distributed operating system for permissioned blockchains[C]//*Proceedings of the Thirteenth EuroSys Conference*. New York: ACM, 2018: 1-15.

[16] Benet J. IPFS-content addressed, versioned, P2P file system [EB/OL]. [2024-01-12]. http://arxiv.org/abs/1407.3561.

[17] Pedersen T P. A threshold cryptosystem without a trusted party[C]//*Advances in Cryptology—UROCRYPT'* 91: *Workshop on the Theory and Application of Cryptographic Techniques Brighton*. Berlin, Heidelberg: Springer-Verlag, 1991: 522-526.

[18] Pedersen T P. Non-interactive and information-theoretic secure verifiable secret sharing[M]//*Advances in Cryptology—CRYPTO* ' 91. Berlin, Heidelberg: Springer-Verlag, 2007: 129-140.

□