electronics

MDPI

Article

An Improved Multi-Authority Attribute Access Control Scheme Base on Blockchain and Elliptic Curve for Efficient and Secure Data Sharing

Ben Xie, Yu-Ping Zhou *, Xin-Yu Yi and Chen-Ye Wang

School of Computer Science, Minnan Normal University, Zhangzhou 363000, China
* Correspondence: yp_zhou@mnnu.edu.cn

Abstract: With the rapid development of Internet of Things technology, sharing data safely and efficiently in different Internet of Things enterprises is becoming increasingly urgent. Traditional schemes usually use third-party centralized cloud storage and a single central authoritative organization to realize data storage and access management during data sharing. However, this centralized scheme design has the potential for a single point of failure. When the cloud storage platform is subjected to malicious attacks, it may lead to data loss or privacy leakage problems. Secondly, there is a trust crisis in the design of authoritative central organizations, and centralized rights management makes the data sharing process opaque. In order to address these shortcomings, an improved blockchain and elliptic curve-based multi-authority attribute access control scheme is proposed. Firstly, the interplanetary file system is used to store the ciphertext of symmetric encryption data to solve data leakage and tampering in centralized cloud storage. Secondly, the elliptic curve cryptography-based improved multi-authority ciphertext policy attribute-based encryption algorithm is used to encrypt the symmetric key. It can solve the single point of failure problem of user attribute management and significantly reduce the attribute encryption algorithm's time and resource consumption. Thirdly, the data-related information is uploaded through the smart contract, and the attribute access threshold is set. Only qualified users can view the private information. Finally, the simulation experiments evaluate the efficiency and effectiveness of the scheme from three perspectives: data storage, smart contract, and attribute encryption.

Keywords: blockchain; data sharing; smart contract; ECC; attribute-based encryption

Citation: Xie, B.; Zhou, Y.-P.; Yi, X.-Y.; Wang, C.-Y. An Improved Multi-Authority Attribute Access Control Scheme Base on Blockchain and Elliptic Curve for Efficient and Secure Data Sharing. *Electronics* **2023**, *12*, 1691. https://doi.org/10.3390/electronics12071691

Academic Editors: Wenquan Jin, Meilan Jiang and Shabir Ahmad

Received: 24 February 2023
Revised: 18 March 2023
Accepted: 28 March 2023
Published: 3 April 2023

1. Introduction

With the rapid progress of Internet of Things (IoT) technology, large and varied amounts of IoT data are generated daily. Enterprises in different IoT systems need to multi-party share data in order to reasonably and fully develop the maximization of data value [1]. However, the data-sharing process will involve some confidential information pertaining to various enterprises. Therefore, it is urgent to construct a convenient and efficient data sharing scheme with high security and privacy protection. In the traditional scheme, the collected data are uploaded to a third-party cloud storage platform, and enterprises interact with each other through a central authority to complete access control [2]. There are hidden dangers of a semi-trusted center and a single point of failure in this centralized shared design system. Therefore, it is necessary to study a new solution to solve the question. As a representative of decentralization, blockchain has always attracted attention and can be used to build a trusted decentralized solution [3].

Blockchain is a distributed ledger technology [4] with traceability, transparency, tamper resistance and decentralization advantages. Because of the unique advantages of blockchain, it can provide high security and traceability for data transactions on the chain, which is an important consideration for multi-party trust cooperation. Currently, blockchain

applications can be seen in all fields [5], mainly through the following four mainstream platforms: Bitcoin, Ethereum, Corda and Hyperledger Fabric. The comparison [6,7] between them is shown in Table 1.

Table 1. Attribute comparison of blockchain platforms.

	Bitcoin	**Ethereum**	**Corda**	**Hyperledger Fabric**
Category	Public Blockchain	Public Blockchain	Distributed Ledger Platform	Consortium Blockchain
Description	Generic blockchain platform	Generic blockchain platform	Financial Industry Special Platform	Modular blockchain platform
Consensus algorithms	POW	POW, POS	Notary mechanism	PBFT
Smart contract	No	Yes (Solidity)	Yes (Kotlin, Java)	Yes (Go, Java)
Privacy	No	No	Yes	Yes
Scalability	No	No	No	Yes
Currency	Yes	Yes	No	No
Identity Authentication	No	No	Yes (digital certificate)	Yes (digital certificate)

Although Bitcoin is the prototype of blockchain technology, it does not support smart contracts and privacy protection, so it cannot be applied to complex business scenarios. As a product of blockchain 2.0, Ethereum is currently the hottest public blockchain system. It can build applications and organizations and hold assets, trade and communication technologies in an environment not controlled by centralized regulators. However, any transaction on Ethereum requires a fee, which is suitable for creating decentralized autonomous organizations. Corda is a technical architecture dedicated to financial services, which abandons block and chain structures and better separates participants' business data. Corda uses the notary mechanism to make the network structure more fixed, but also loses flexibility and scalability. Fabric, as the representative of the consortium blockchain, is an open-source distributed ledger platform for enterprise application development. It modularizes technologies such as rights management, authentication, and consensus mechanisms to support pluggability and expansion. Fabric realizes different business logic by designing and developing smart contracts, which can be more conveniently applied to complex environments. It is the first choice for enterprise development blockchain projects. Therefore, we use Fabric as the underlying blockchain platform to research and develop efficient and secure data sharing schemes in the IoT. However, each node in the blockchain system must maintain a complete ledger of all transactions on the chain. If a mass of data are directly uploaded to the blockchain, each node will maintain these data, which may cause system congestion and bring high consumption and load to the client. Therefore, it is necessary to solve this problem through off-chain storage and on-chain retrieval. The Interplanetary File System (IPFS) [8] is a distributed system based on content storage. A mass of original data are stored in IPFS, and an on-chain data retrieval table is constructed using the addressing hash returned by IPFS and uploaded to the chain. This reduces the network load and storage pressure of the blockchain and avoids the problem that storing data on a third-party semi-trusted platform may lead to data leakage or loss [9].

An important problem to be solved in data sharing is how to carry out safe and efficient access control. Simply speaking, it refers to how data owners control and manage their data reasonably and effectively and how to distinguish users who can access their data. In many current studies, Ciphertext Policy Attribute-Based Encryption (CP-ABE) technology is considered one of the efficient and secure data protection methods suitable for complex IoT environments because it can achieve fine-grained one-to-many access control [1]. Data owners encrypt data by setting appropriate access policies. The ciphertext can be decrypted when the user attributes satisfy the access policy. However, in the traditional CP-ABE scheme [10], the user-based attribute authentication and key distribution are entirely managed by a central authority, which poses the risks of an opaque authentication process.

Therefore, in order to solve this problem, Allison and Waters proposed a decentralized attribute-based encryption scheme [11] in which attribute assignment to users is managed by multiple attribute authorities. In subsequent research, the scheme has been continuously improved, greatly reducing the possibility of problems in attribute authentication and key distribution [12]. These studies also ignore the control of the data owner. Therefore, in this paper, the data owner strengthens the control of the data by setting the corresponding attribute security threshold in the policy when uploading the data. However, in the CP-ABE encryption scheme, the computing resources and time consumed in the encryption and decryption process are also essential factors to consider. In this paper, an improved Multi-Authority Ciphertext Policy Attribute Based Encryption (MA-CPABE) algorithm [13] based on Elliptic Curve Cryptography (ECC) is applied. The fast and straightforward scalar multiplication on ECC is used to replace the traditional complex bilinear pairing operation, improving the algorithm's security and dramatically reducing resource consumption and calculation time.

Therefore, to solve some of the above problems and provide a safe and efficient data sharing scheme for the IoT system, an improved MA-CPABE base on blockchain and ECC is proposed in this paper. Firstly, the data owner symmetrically encrypts the original data and then stores the obtained ciphertext in the distributed storage platform, IPFS. After the storage is completed, IPFS will return the content addressing the hash of the data ciphertext. The owner uses the attribute-based encryption algorithm to encrypt the symmetric key. Then, the owner manages the data by using the smart contract deployed on the blockchain to upload the relevant data information and setting the corresponding attribute threshold. Data visitors need to use multiple attribute agencies to jointly generate attribute tokens for themself and utilize smart contracts to access data. This scheme solves the problem of how to share data safely and efficiently by means of distribution in a complex IoT environment.

The contributions of this article are summarized as follows.

1. In this paper, IPFS is used as a distributed storage platform, which not only achieves on-chain retrieval and off-chain storage, but also solves the shortcomings of privacy leakage, single point of failure and repeated storage (IPFS automatically deletes duplicate data content) in centralized storage mode.
2. A data sharing scheme based on consortium blockchain and improved attribute encryption is proposed. It solves some problems in the past scheme with the idea of distribution, and the blockchain can provide auditable action logs to make the data sharing process more transparent.
3. The MA-CPABE encryption algorithm improved by ECC is adopted. It solves the problem of attribute distribution, which depends on the centralized third party and is opaque in the traditional attribute encryption algorithm. Moreover, it reduces time and resource consumption in encryption and decryption.
4. Use Hyperledger Fabric chaincode technology to realize data upload, query and access. Only users who meet the access control conditions set by the data owner can view the privacy information of the data.

The chapter arrangement of this paper is organized as follows. Some related research on access control and blockchain is introduced in Section 2. The relevant background knowledge of this paper is introduced in Section 3. The system model of the data sharing scheme is introduced in Section 4. The detailed data sharing process and corresponding algorithms are introduced in Section 5. The security, function and experimental analysis of the scheme are analyzed in Section 6. Finally, a general conclusion of the full text is given in Section 7.

2. Related Work

In this section, some of the existing research about data sharing is introduced, mainly covering two aspects: access control and blockchain.

There has been much research exploring how to conduct secure access control. As a traditional access control method, Discretionary Access Control (DAC) [14] is centered

on the data owner. It sets the Access Control List (ACL) according to its wishes to decide whether to grant access to other users. This method requires users to maintain their ACL to manage data. However, in the complex IOT environment, users in the system may need to change their permissions frequently, which cannot be dealt with in time. The Role-Based Access Control (RBAC) [15] is a scheme that associates roles and permissions. Each user has its role attributes in the system, and each role has different permissions. Compared with DAC, RBAC simplifies user rights management, but it does not provide fine-grained access control. Attribute-Based Access Control (ABAC) [16] manages and controls information according to user attributes in the system, and only the user who meets the relevant attribute requirement can access the information. ABAC provides secure and efficient data protection in complex IoT environments as a flexible fine-grained access control method.

The CP-ABE [10] proposed by Bethencourt et al. is a classical encryption scheme in ABAC. In this scheme, the sender constructs the access policy and embeds it into the ciphertext; only when the receiver's attributes meet the attribute requirements in the access policy can the decryption be successful. However, centralized management is adopted for attribute authentication and key distribution, which makes the access control process opaque. In 2007, Chase [17] proposed the MA-ABE scheme to improve traditional attribute-based encryption. Each user in the system has a unique identifier representing their identity, and multiple attribute authorities are set up to assign attributes to users. On this basis, Allison and Waters further improved the MA-CPABE scheme [11]. By introducing a linear secret sharing scheme and using a conversion algorithm, the access policy tree composed of AND and OR thresholds was converted into an access matrix, enhancing the access policy expression ability. Thus, it was more suitable for distributed networks. However, in the above attribute encryption scheme, bilinear pairing completes the encryption and decryption process. Bilinear pairing is computationally expensive and sometimes unacceptable for devices with limited resources. In 2022, Sandhia and Raja proposed an MA-CPABE-ECC scheme [13] for data sharing in the cloud. In this scheme, complex bilinear pairing is replaced by simple scalar multiplication in ECC in the scheme, which has a smaller key and reduces the calculation time.

Although access control can ensure data security in the process of data sharing, centralized management and the inability to provide complete access control records still hinder the development of data-sharing research. With the emergence of a decentralized representative blockchain, another way of thinking has been brought to data sharing research. A trusted multi-party cooperation platform can be built through blockchain. Yang et al. [18] proposed an overall architecture for data sharing and transactions, which guarantees the security of the entire process through the distributed and transparent characteristics of the blockchain and prevents data from being tampered with during storage through encryption algorithms. Guo et al. [19] proposed a scheme to support data sharing with blockchain as the underlying platform for the complex scenarios of the IoT. The scheme solves the blockchain storage problem through off-chain storage and on-chain search. The original data are stored in the database under the chain, and the summary information of the data is uploaded to the chain. Users query and verify the data through blockchain. Alshalali et al. [20] proposed a scheme for sharing electronic medical records based on Fabric. The hospital stores the patient's electronic medical record data and the user ID authorized to access the medical record in the database. When the visitor needs to view the medical record data, the blockchain is used to verify whether its ID is authorized to access. Chen et al. [21] proposed a Fabric-based cross-enterprise data sharing scheme. The data owner and user build a communication platform through the blockchain and complete the entire data sharing process through on-chain transactions. In addition, IPFS is introduced to reduce the storage pressure of the blockchain, and the data parties use the elliptic curve digital signature algorithm multiple times to ensure the data security of transactions on the chain.

Liu et al. [22] proposed a scheme that combined ABAC with blockchain technology. It solves the problem that one-to-many fine-grained access control is difficult to achieve using the traditional scheme. However, the data are uploaded to the cloud without encryption in this scheme. Because the third-party cloud storage service is semi-trusted, the integrity and privacy of the data on the cloud cannot be guaranteed. Lu et al. [23] used blockchain to build a platform, introduced IPFS instead of cloud storage, and used symmetric encryption and CP-ABE algorithms to achieve access control of data. Liang et al. proposed the PDPChain scheme [24], which uses the improved Paillier homomorphic encryption to improve data availability and enable data management. Feng et al. [2] proposed an outsourced parallel computing ABEM-POC model considering the large computing time and resource consumption in the CP-ABE process. The most time-consuming and resource-consuming part of the CP-ABE is transferred to the external edge computing platform, which greatly accelerates the calculation speed and reduces consumption. Many references [25–28] apply CP-ABE to blockchain and propose corresponding architectures and algorithms.

However, attribute management is achieved through a centralized authority in the above schemes. To solve this problem, Guo et al. [29] proposed a blockchain-based MA-CPABE scheme. Users call smart contracts through the API interface to collect identity attributes issued by multiple attribute agencies. The decryption key can be obtained through the smart contract when the user attribute meets the access policy. Finally, the shared data can be decrypted by using the key. However, each attribute agency only manages one attribute in the scheme. Sammy et al. [30] used improved hierarchical attribute access control and outsourced decryption to allow multiple authorities to provide dynamic attributes to data requesters. Qin et al. [31] combined a smart contract and a Shamir secret sharing scheme to enable multiple authorities to jointly manage user attribute authentication. At the same time, four smart contracts were designed to achieve attribute publishing, collection and key generation.

Blockchain technology provides traceable and transparent transaction records to ensure that data will not be tampered with and solve the trust problem of data sharing under centralized management. Access control technology authorizes users in the system to access data. In this paper, an improved MA-CPABE base on blockchain and ECC is proposed, and some smart contracts are designed to achieve data upload, query, and access. Multiple attribute authorities can jointly manage user attributes via blockchain, which makes the data sharing transparent and auditable.

3. Preliminaries

In this section, first, the basic definitions of ECC and linear secret sharing scheme are introduced. Then, the Fabric blockchain, IPFS and MA-CPABE schemes used in this scheme are introduced.

3.1. ECC

ECC [32] is asymmetric encryption based on elliptic curve discrete logarithm problem. The most basic definition of the elliptic curve equation in ECC is as follows.

$$y^2 = x^3 + ax + b, \quad 4a^3 + 27b^2 \neq 0 \tag{1}$$

Define an elliptic curve E of order q in a finite domain $GR(q)$. Moreover, G is a generator with the order r; for any point Q on the E, it can be calculated by $Q = kG, k \in Z_r$.

There are three main steps involved in ECC encryption. First, the plaintext information that needs to be encrypted is mapped to point Q on the E. Then, the encryptor (Alice) and the decryptor (Bob) perform the following three steps.

1. Key generation.
   (a) Alice and Bob select an elliptic curve with the same parameters $y^2 = x^3 + ax + b(mod\ p)$ and point G as the generator.

(b) Alice randomly selects an integer $S_a \in Z_p$ as the private key and then computes the corresponding public key $P_a = S_a \cdot G$.
(c) Bob randomly selects an integer $S_b \in Z_p$ as the private key and then computes the corresponding public key $P_b = S_b \cdot G$

2. Encryption
Encrypt the information mapped to point Q. Alice calculates the ciphertexts $C_1 = kG$ and $C_2 = Q + k \cdot P_b$, where $k \in Z_p$ is an integer randomly selected by Alice. Alice sends the calculated ciphertexts to Bob.
3. Decryption
After receiving the ciphertexts, Bob uses his private key S_b to compute the point Q according to the formula $C_2 - S_b C_1$.

$$C_2 - S_b C_1 = (Q + kP_b) - S_b(kG) = (Q + kS_b G) - S_b kG = Q \quad (2)$$

Then, he obtains plaintext information by mapping point Q back to E.

3.2. Linear Secret Sharing Scheme (LSSS)

Secret sharing scheme [33] divides a secret into n different parts and then distributes it to all parties. A group of authorized parties can recalculate the secret by combining their secrets.

The specific introduction of LSSS is as follows.

1. The secret shares of the parties form the vector on Z_p.
2. The shared matrix A is composed of n rows and m columns. For $i \in 1, \ldots, n$ each line i marked with a function ρ is associated with one of the parties. Suppose $s \in_p$ is the secret to be shared. The first element of the column vector v is s, and the remaining elements need to be randomly selected from Z_p. Then, $A \cdot s$ is computed as the sharing vector of the secret s, where vector $v = (s, r_2, \ldots, r_m)$, and $r_2, \ldots, r_m \in Z_p$.
3. Suppose an arbitrary authorization set $S \in T$, where T is a self-defined access policy tree. $\{c_i \in Z_p\}_{(i \in 1,\ldots,n)}$ is a constant set, compute the original secret $s = \sum_{(i \in 1,\ldots,n)} \lambda_i c_i$, where λ_i is the share of secret s.

The A can be generated by the conversion algorithm. The monotone Boolean formula is used as the input to access the structure tree. Each leaf node represents an attribute, and non-leaf nodes are AND and OR thresholds. The A is the output; each row in A represents an attribute. The conversion algorithm is as follows.

1. If the parent node is an OR threshold marked with vector v, then its two child nodes are represented by vector v, and the value of the counter V is unchanged.
2. If the parent node is an AND threshold marked with vector v, nought is filled at the end of the vector v to make the length of v equal to the counter value. Then, the left node is marked with $(0, \ldots, 0) \parallel -1$, where the number of zero is V, and the right node is marked with $v \parallel 1$. Finally, the value of the counter V is increased by one.

Figure 1 shows how the attribute access structure tree is transformed into an access matrix by the transformation algorithm. The generated LSSS access matrix A is as follows.

$$A = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{matrix} \rho(1) = a \\ \rho(2) = b \\ \rho(3) = c \\ \rho(4) = d \end{matrix} \quad (3)$$

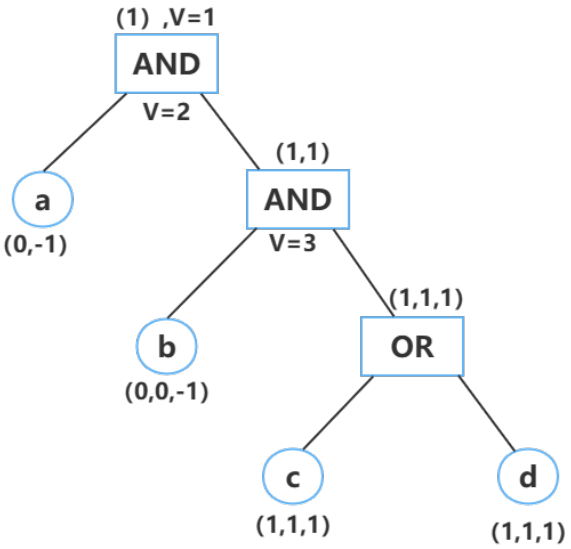

Figure 1. Accessing structure tree to generate LSSS matrix.

3.3. Hyperledger Fabric

Blockchain consists of multiple blocks connected by address hash values to maintain a transparent and immutable ledger in distributed point-to-point networks. Any transactions in the blockchain network will be recorded in the ledger. The scheme proposed in this paper is based on Hyperledger Fabric [34].

Identity management: The authentication, authorization and management of nodes in Fabric are completed by the Member Service Provider (MSP). As the default certification authority (CA) of Fabric, CA needs to issue identity certificates for each node. Each new member node must be authenticated by the CA before joining the current blockchain network.

Smart contract: Smart contract is a contract that uses computer language to replace language to record terms and is automatically executed by a program. In other words, the smart contract is a digital version of a traditional contract that runs on blockchain networks and is automatically executed by programs. The smart contract in Fabric is called chaincode, and users can invoke the chaincode-related API to access, modify and create a set of key-value pairs in the ledger.

Transaction: Transactions are generated by the user implementing the chaincode on the client application side to read, modify and write data on the ledger. Each transaction is approved after endorsement and consensus within the network.

Ledger: The ledger of Fabric consists of block log and world state. The transaction ledger needs all nodes in the Fabric channel to maintain together. When a transaction is completed, the latest states (current value) of all key-value pairs in the current blockchain are recorded in world state. Block log means adding all generated transactions as blocks to an immutable chain.

3.4. Interplanetary File System (IPFS)

IPFS is a point-to-point storage system based on file content [35]. IPFS will return a unique addressing hash for files stored in the system. Nodes in IPFS are the same as blockchain, and multiple nodes maintain the same storage network. Problems with a single node do not destroy the entire network, so there is no single point of failure risk as there is in traditional networks. Because IPFS is based on storing data content, the same data will be stored only once, so IPFS can avoid data redundancy and reduce storage space.

3.5. MA-CPABE Scheme

A classical MA-CPABE scheme is composed of four algorithms [11]: Setup (system setup and attribute authority setup), Encrypt, KeyGen and Decrypt.

$System\_Setup(q \rightarrow PP)$: The CA executes the algorithm to initialize the entire system. The CA takes a large prime number q as the input of security parameters and runs the algorithm to determine the public parameters of the system PP.

$Authority\_Setup\{(PP \rightarrow (PK, SK)\}$: The algorithm is executed by various attribute authorities. Attribute authority inputs the public parameter PP and outputs its attribute public key PK and private key SK.

$KeyGen\{(PP, i, SK, GID) \rightarrow SK_{i,GID}\}$: The attribute authority takes the public parameter PP, the user unique identifier GID, the attribute i and the attribute private key SK of the corresponding attribute of the attribute institution as input. It outputs the attribute key $SK_{i,GID}$ of attribute i, corresponding to the user identity GID.

$Encrypt\{(PP, (A, \rho), M, \{PK\}) \rightarrow CT\}$: The user inputs the information M that needs to be encrypted, PP, the corresponding access policy matrix (A, ρ) and the public key set $\{PK\}$ of the attribute authority. The output ciphertext CT is encrypted by the encryption algorithm.

$Decrypt\{(PP, CT, \{SK_{i,GID}\}) \rightarrow M\}$: The user inputs public parameters PP, ciphertext CT and attribute key set $\{SK_{i,GID}\}$. Plaintext M can be decrypted successfully if the user attribute meets the access policy in the ciphertext.

4. System Model

4.1. System Architecture

The overall architecture of the cross-enterprise data sharing solution in the IoT, as shown in Figure 2, consists of five layers: data collection, user, storage, interaction and access control. The data collection layer comprises IoT devices managed by IoT enterprises, such as smartphones, electronic probes, drones, etc. These devices are the primary source of IoT data. Enterprises share the data generated by these IoT devices reasonably and safely to achieve win–win cooperation. The user layer is the enterprise users in the scheme, including data owners and visitors, and is the most crucial entity in the data sharing process. The storage layer is composed of IPFS, which is the main carrier for storing data in the scheme. The interactive layer is the Fabric blockchain network and the underlying platform of the system solution. All operations in the data sharing process need to be completed in the form of on-chain transactions. The access control layer is composed of a certificate authority and multiple attribute authorities, and its primary function is to realize the management of data access rights by system users. The focus of secure access control is key issuance and user attributes authentication in the scheme.

4.2. System Model

There are six entities in the system model: CA, IPFS, attribute authorities, data owner, data visitor and Fabric blockchain.

CA: As an MSP entity in Fabric, CA is responsible for registering and issuing identity certificates for users in the Fabric blockchain. It initializes the system by setting security parameters and collecting the public key of attribute authorities.

IPFS: As the main container for storing data in the system, IPFS provides off-chain data storage services to solve the problem of insufficient storage capacity on the chain; that is, to achieve off-chain storage and on-chain retrieval.

Attribute Authorities(AAs): As the authority of attribute authentication in the system, AA is mainly responsible for the attribute authentication of users in Fabric and publishes the corresponding attribute sub-keys to users through blockchain. Each attribute is jointly managed and authenticated by multiple AA. Similarly, each AA needs to be responsible for multiple attributes. In short, there is a many-to-many mapping between attribute authorities and attributes. The relationship diagram is shown in Figure 3.

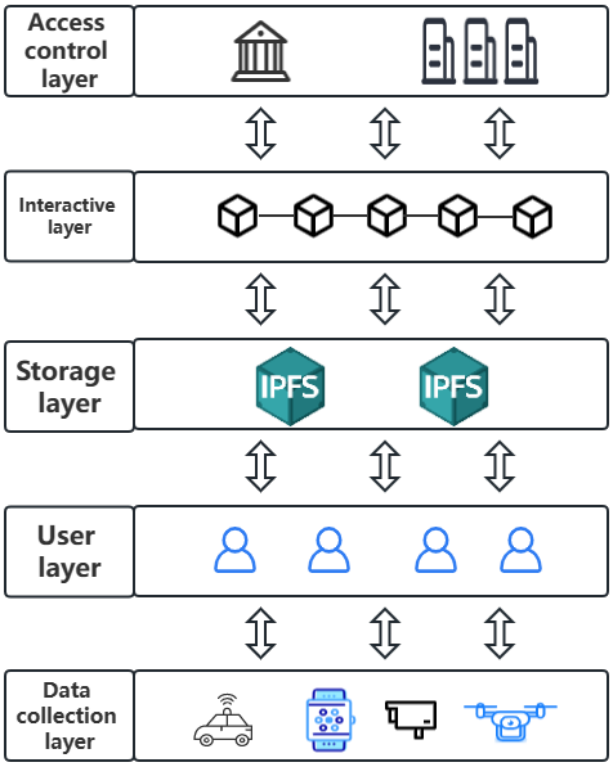

Figure 2. System architecture.

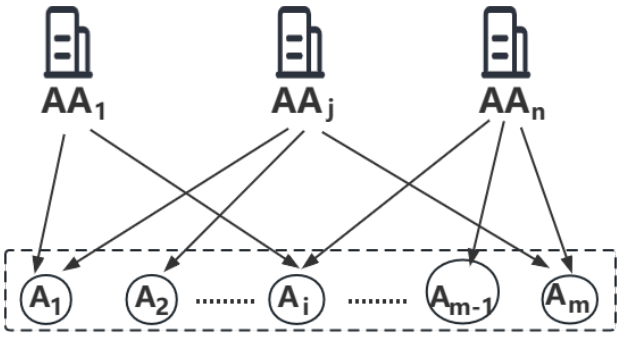

Figure 3. The mapping between AAs and attributes.

Data Owner (DO): DO is the owner of data resources, which protects data through two encryption algorithms. The ciphertext after attribute encryption is used to construct a Data Element Table (DET), and the smart contract is implemented to upload the DET. DO enforces secure access control of data by setting access policies and attributes security thresholds.

Data Visitor (DV): DV is a user who wants to access and use data from the DO. DV requests the decryption key related to its identity attribute through the blockchain. Data can be successfully accessed and decrypted if the DV attribute meets the access policy of ciphertext.

Fabric Blockchain: As the underlying platform of the system solution, Fabric is mainly responsible for storing relevant data information, completing data sharing operations through relevant smart contracts, and providing a transparent and auditable ledger for on-chain transactions.

As shown in Figure 4, the data sharing system model proposed in this paper is as follows.

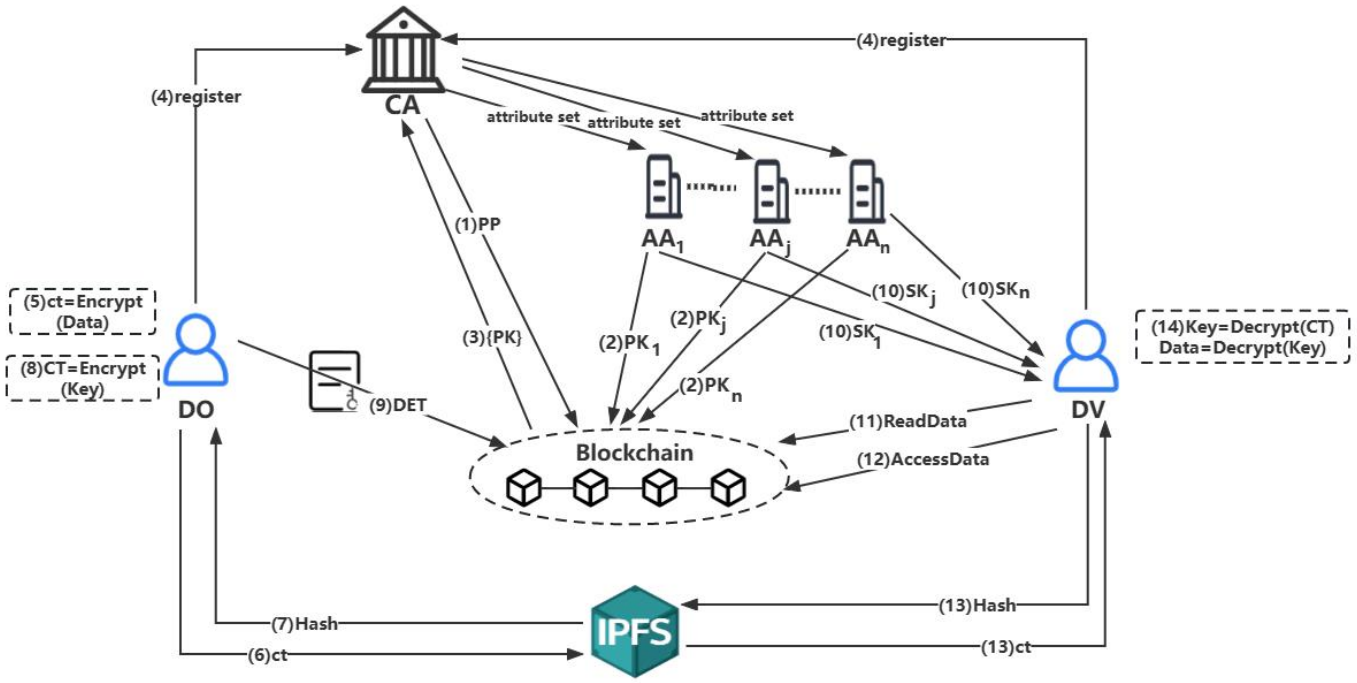

Figure 4. System model of data sharing scheme based on attribute encryption in Fabric.

1. CA generates the public parameter PP and publishes it to the blockchain. Moreover, it distributes different attribute sets for each attribute authority.
2. Each attribute authority AA_j generates a key pair (PK_j, SK_j) based on the attribute it manages and uploads its public key PK_j to the blockchain.
3. CA collects all PK_j to the attribute public key set $\{PK\}$ and publishes it to the blockchain.
4. When system users such as DO and DV apply to join the blockchain, they need to register with the CA to obtain the corresponding digital certificate and system parameters (including the attribute public key set $\{PK\}$ and public parameter PP).
5. DO encrypts data with the symmetric encryption algorithm AES to obtain ciphertext ct, where the key of the symmetric encryption algorithm AES is represented by Key.
6. DO stores ct in IPFS.
7. IPFS returns addressing hash $Hash$ to DO.
8. DO constructs the access policy (A, ρ) of the data and encrypts Key by using the MA-CPABE to obtain ciphertext CT.
9. DO constructs a DET based on various pieces of information about the data, and each DET corresponds to the identity ($DataID$) of the data on the blockchain (one $DataID$ corresponds to a unique DET). The smart contract **UploadData** is called to upload the DET and the corresponding attribute threshold T in the access policy.
10. DV applies for attribute sub-tokens to AA through blockchain. Attribute authority AA_j issues attribute sub-token $SK_{i,j}$ to DV according to the identity of DV and the attribute at managed by them-self.
11. DV calls the smart contract **ReadData** to query part of the data information of all DETs in the current blockchain ledger, such as data summary, data size, data identity, etc. DV determines which data it needs using the descriptive information of the data.
12. DV determines the needed data, calls the data access smart contract **AccessData** according to the $DataID$ of the data, and uploads its attribute sub-token set. If the attribute sub-token set uploaded by the DV satisfies the access policy and attribute threshold set by DO, the DET of the data can be successfully accessed, and the decryption key SK will be generated.

13. DV retrieves the ciphertext ct stored in IPFS according to the data hash address $Hash$ in DET.
14. DV uses SK to decrypt the ciphertext CT in DET to obtain the symmetric key Key. Then, DV uses Key to decrypt the ciphertext ct to obtain the original data information $Data$.

5. Scheme Overview

5.1. Scheme Specific Process

There are four stages included in the scheme: System Initialization, Data Encryption, Data Access and Data Decryption. Each stage and the algorithm used will be described in detail below.

5.1.1. System Initialization

The specific process of the System Initialization phase is shown in Figure 5.

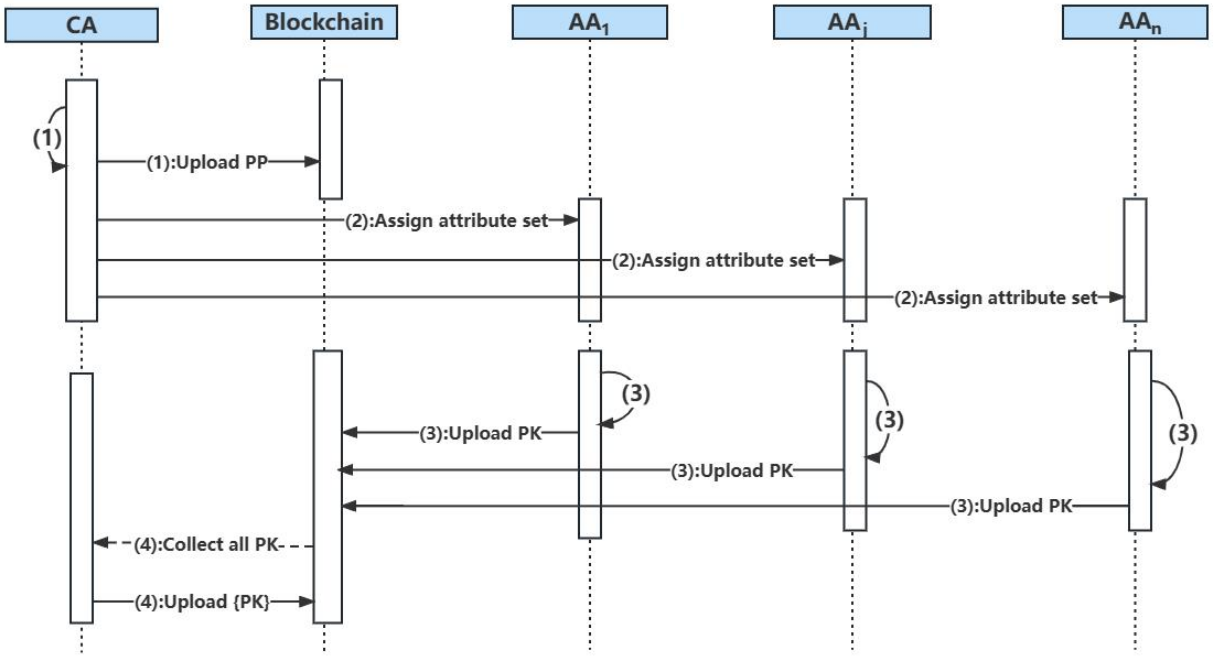

Figure 5. System initialization flow chart.

1. CA runs the system initialization algorithm $System\_Setup(q \rightarrow PP)$, in which the security parameter q is entered, and the global parameter PP of the system is obtained and published to the blockchain. The details are as follows.
   The initialization algorithm of the system is to input a large prime number q as the security parameter and output the public parameter $(q, GF(q), E, G, r, h, H)$ of the system. E is an elliptic curve in the finite domain $GF(q)$ of the q order finite field. A point G on the r order cyclic group GR containing all points on E is selected as the generator, and all points in GR are generated by $G \cdot r$, where $i < r$. The positive integer h is an auxiliary factor mainly used to calculate $h \cdot r = |E|$. Each user in the system has a unique identifier GID, and the user identifier GID is mapped to the element of Z_r through the hash function $H : \{0,1\}^* \rightarrow Z_r^*$.
2. CA randomly assigns attributes to AA, and each attribute needs to be jointly managed by x attribute authorities. Each AA obtains different attribute sets for management.
3. AA run algorithm $Authority\_Setup\{(PP \rightarrow (PK, SK)\}$. Each AA generates its key pairs (PK_j, SK_j) according to the attribute set $AA_j.AttSet$, then uploads its public key PK_j to the blockchain. The algorithm details are as follows.

Each AA randomly selects an integer $n \in Z_r$ as its private key and the corresponding public key $n \cdot G$ is calculated. For each attribute A_i in the system, the AA randomly selects an integer $a_i \in Z_r$ as the private key and $PK_i = a_i \cdot G$ as the public key.

4. CA collects the public key PK_j uploaded by all AA, aggregates these attribute public keys into a public key set $\{PK\}$ and publishes it to the blockchain.

5.1.2. Data Encryption

The specific process of the Data Encryption phase is shown in Figure 6.

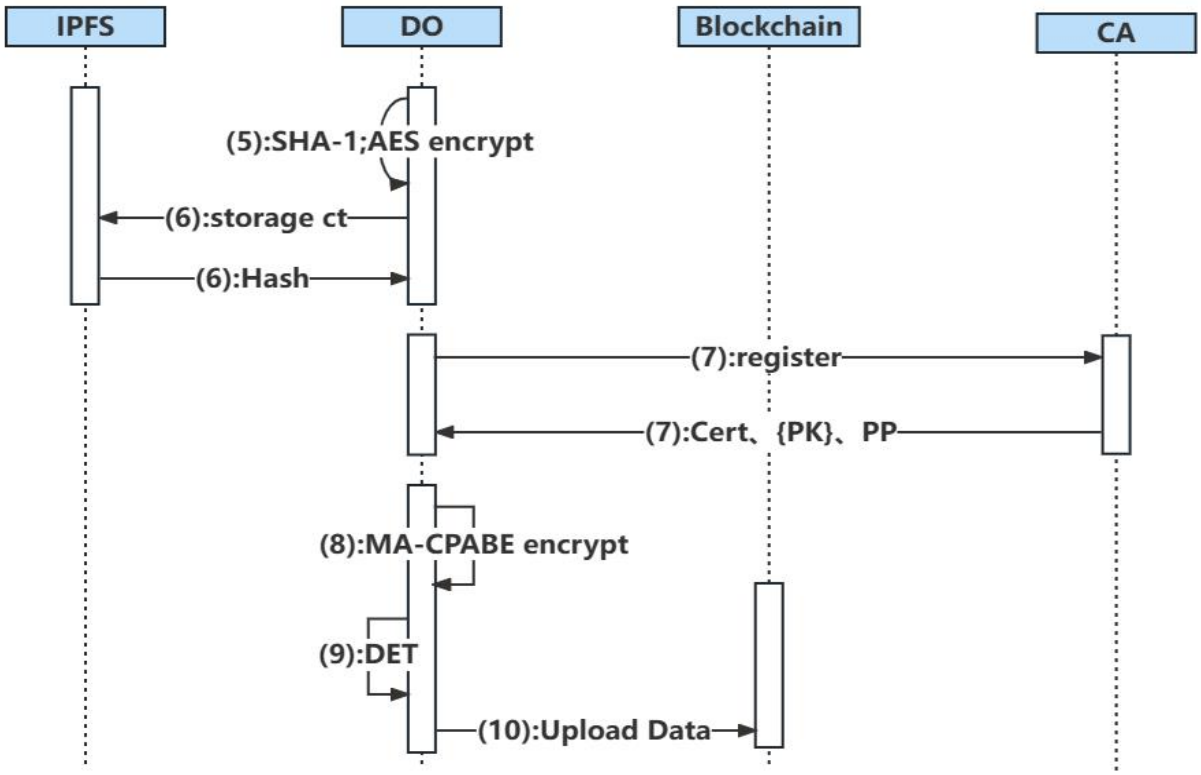

Figure 6. Data encryption flow chart.

5. DO first computes a hash value *hash* for the *Data* with the SHA-1 algorithm and then encrypts *Data* to obtain ciphertext *ct* with AES algorithm.
6. DO stores the encrypted ciphertext *ct* in IPFS. Then, IPFS returns the corresponding address hash value *Hash*.
7. DO applies for permission to join the blockchain network through CA. If the application is accepted, PP, digital certificate $Cert_{DO}$ in the blockchain and attribute public key set $\{PK\}$ will be returned.
8. DO constructs the access policy (A, ρ) and executes the encryption algorithm $Encrypt\{(PP, (A, \rho), Key, \{PK\}) \rightarrow CT\}$ to encrypt the AES key *Key* to obtain the key ciphertext CT. The algorithm is as follows.
   (a) First, it maps the key *Key* to be encrypted to point M on the elliptic curve E, then it randomly selects an integer $s \in Z_r$ and calculates $C_0 = M + sG$.
   (b) It takes the access policy set by DO as the input, then outputs the access matrix A of n × l and maps the attributes in the access policy to matrix A using the function ρ.
   (c) It selects vectors $v = (s, v_2, \ldots, v_m) \in Z_r$ and $u = (0, u_2, \ldots, u_m) \in Z_r$ at random and calculates $\lambda_x = A_x \cdot v$ and $\omega_x = A_x \cdot u$, respectively, where A_x represents the x-th row of the matrix A.

(d) Finally, the ciphertexts are calculated by $C_{1,x} = \lambda_x \cdot G + \omega_x \cdot PK_{\rho(x)}$ and $C_{2,x} = \omega_x \cdot G$, respectively.

9. DO constructs DET based on information such as ciphertext CT and data hash address $Hash$. The DET details are shown in Figure 7.
Data description: Descriptive information of data.
DataID: The unique identifier of data on the blockchain, including the enterprise name and number. This is the basis for querying the data information. Holder: Enterprise name of the data owner. DataSummary: Summary description of data information. ID: The unique identifier of DO in the blockchain is the blockchain address. Sign: Digital signature of DO. Size: The size of data.
Data privacy: Data privacy information.
Hash: The data address returned by IPFS and the only basis for querying data. CT: The ciphertext AES key is encrypted with MA-CPABE. hash: The hash of data plaintext is the basis for checking data integrity.
10. DO uploads the DET to the blockchain through the smart contract **UploadData** (see Section 5.2.1 for more information) and sets the access attribute threshold t for the data.

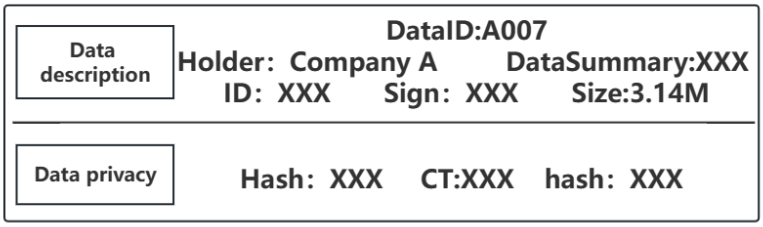

Figure 7. Data element table.

5.1.3. Data Access

The specific process of the Data Access phase is shown in Figure 8.

11. DV applies to join the blockchain. CA agrees to return DV's digital certificate $Cert_{DV}$ in the blockchain, which includes DV's unique identifier VID in the system, as well as the global parameter PP and attribute public key set $\{PK\}$ that needs to be used in the subsequent decryption process.
12. DV applies for attribute identity through blockchain. AA run algorithm $KeyGen\{(PP, i, SK, GID) \rightarrow SK_{i,GID}\}$ in the system to issue attribute sub-token $SK_{i,j,VID}$ according to DV identity. The algorithm is as follows.
The AA generates the attribute key of attribute A_i for users with VID. The key calculation method is as follows.

$$SK_{i,VID} = a_i + H(VID) \cdot n \tag{4}$$

13. DV uses the smart contract **ReadData** to query information of the data description part in all DET (see Section 5.2.2 for more information). Then, it finds the needed data according to the returned data description information, where $DataID$ is the basis for subsequent query.
14. The DV uploads its own set of attribute tokens $\{SK_{i,j,VID}\}$ and invokes smart contract **AccessData** to query all the corresponding information based on the $DataID$ (see Section 5.2.3 for details). If the attribute security threshold t set by DO is satisfied, the DET of the data can be successfully accessed, and the attribute decryption key $\{SK_{i,VID}\}$ is generated.

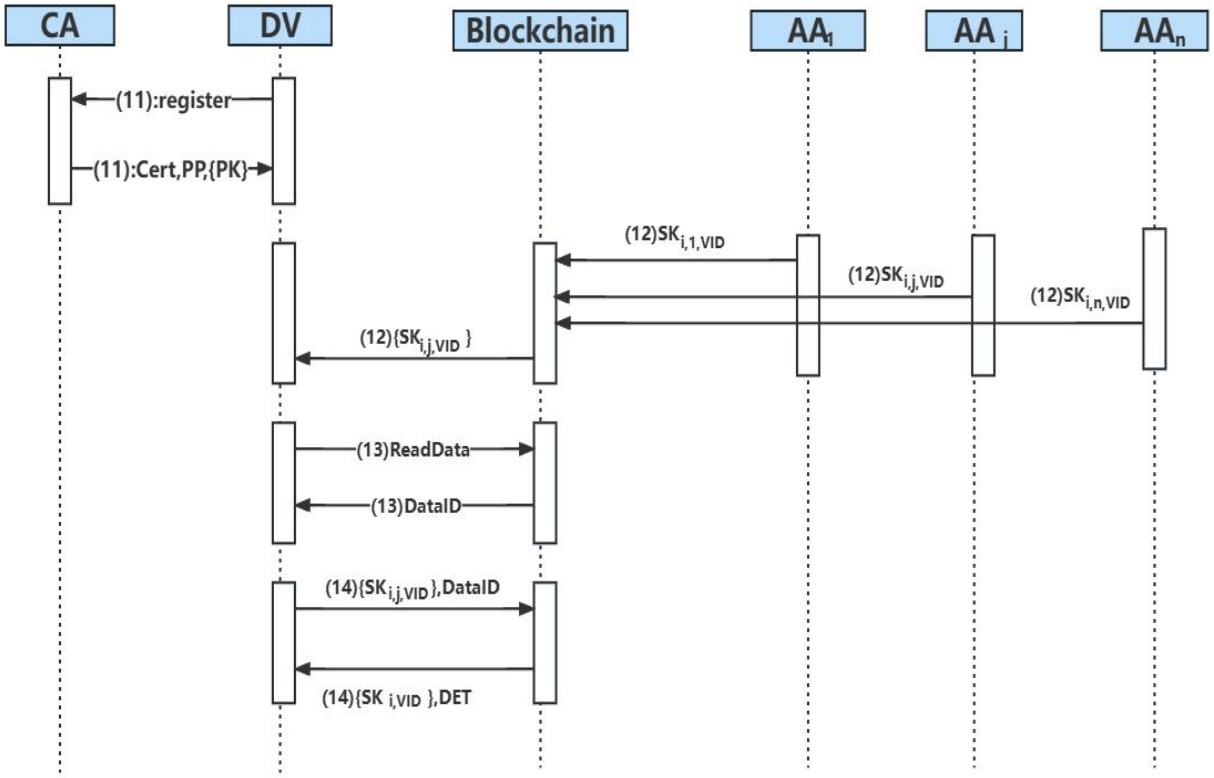

Figure 8. Data encryption flow chart.

5.1.4. Data Decryption

15. DV Downloads DET and the decryption key $\{SK_{i,VID}\}$, and retrieves the ciphertext ct from the IPFS according to the storage address $Hash$ in the DET.
16. DV runs the decryption algorithm $Decrypt\{(PP, CT, \{SK_{i,GID}\}) \rightarrow Key\}$ to obtain the decryption key Key of AES. The algorithm is as follows.
To successfully decrypt the ciphertext, the DV first needs to find a suitable set of rows A_x in matrix A, where $(1, 0, \ldots, 0)$ needs to be in these rows; submits its identity identifier VID and $(C_{2,x}, \rho(x))$ of each line x; and then performs the calculation.

$$\begin{aligned}\sum (C_{2,x} \cdot SK_{\rho(x),VID}) &= \sum (\omega_x G \cdot (a_{\rho(x)} + H(VID)n)) \\ &= \sum (\omega_x a_{\rho(x)} G + \omega_x H(VID)nG) \quad (5)\end{aligned}$$

Based on the above results, it is necessary to continue calculating.

$$\begin{aligned}&\sum C_{1,x} - \sum (C_{2,x} \cdot SK_{\rho(x),VID}) \\ &\quad = \sum (\lambda_x \cdot G + \omega_x \cdot PK_{\rho(x)}) - \sum (\omega_x a_{\rho(x)} G + \omega_x H(VID)nG) \\ &\quad = \sum (\lambda_x \cdot G - \omega_x H(VID)nG) \quad (6)\end{aligned}$$

The DV selects integer $c_x \in Z_r$, which satisfies $\sum c_x A_x = (1, 0, \ldots, 0)$, and performs the computation.

$$\sum c_x (\lambda_x G - \omega_x H(VID)nG) = sG \quad (7)$$

When $v \cdot (1, 0, \ldots, 0) = s$ and $u \cdot (1, 0, \ldots, 0) = 0$. DV decrypts the point M.

$$C_0 - sG = M \quad (8)$$

Last maps M back to the AES key Key.

17. DV uses the symmetric key Key obtained by the attribute decryption algorithm to decrypt the data ciphertext ct.
18. DV hashes the data $Data$ to obtain $Data_{hash}$ using SHA-1 and compares the calculated hash value with $hash$ in DET. The data remain the same if the two hashes are the same.

5.2. Contract Introduction

There are three smart contracts applied in the data sharing scheme: **UploadData**, **ReadData** and **AccessData**. In this section, we describe the proposed smart contract in the form of pseudocode and give the corresponding analysis.

5.2.1. UploadData

The data upload contract is used by DO to upload DET to the blockchain. In order to reduce the storage pressure of the blockchain, DO uses the information of the data to construct the DET, and $DataID$ is used as the retrieval key of DET on the blockchain. Algorithm 1 gives the specific process of storing data information DET. First, lines 1–3 are used to judge whether the contract input parameters are correct. Then, lines 4–7 determine whether $DataID$ is reused to ensure only one $DataID$ corresponds to each DET. Lines 8–10 perform the storage operation and DET is stored in the blockchain as a key-value pair. Lines 11–17 are used to set the threshold of related attributes in the access control policy. If all of the above actions are successful, the DET of the data information is successfully uploaded into the blockchain.

Algorithm 1 UploadData()

Input: DataId,Holder,ID,Sign,DataSummary,Size,Hash,CT,hash,T
Output: bool

1: **if** $\{DataId == null \parallel Holder == null \parallel ID == null \parallel Sign == null \parallel DataSummary == null \parallel Size == null \parallel Hash == null \parallel CT == null \parallel hash == null \parallel T == null\}$ **then**
2: **return** Error("args error")
3: **end if**
4: $Data \leftarrow APIGetStub.GetState(DataId)$
5: **if** $Data! = null$ **then**
6: **return** Error("DataId's DET already exist")
7: **end if**
8: $DET \leftarrow \{DataId, Holder, ID, Sign, DataSummary, Size, Hash, CT, hash\}$
9: $DETjosn \leftarrow json.Marshal(DET)$
10: $APIGetStub.PutState(DataId, DETjson)$
11: **for** $(i = 0 \rightarrow m - 1)$ **do**
12: **if** $A[i]$ in (A, p) **then**
13: $A[i].threshold \leftarrow T$
14: **else**
15: $A[i].threshold \leftarrow 0$
16: **end if**
17: **end for**
18: **return** (true) //Data Upload success

5.2.2. ReadData

The data search contract displays partial information of all DETs in the current blockchain to users, and DV calls the contract to search for the needed data. The specific process of the contract is given in Algorithm 2. The first line is to iterate through the entire blockchain ledger. Lines 2–11 store the data description part in DET in the $DETpart$. Finally, all $DETpart$ are returned.

Algorithm 2 ReadData()

Input: API
Output: $All\ DETpart$
1: $ALLIterator \leftarrow APIGetStub.GetStateByRange()$
2: **for** $(AllIterator.HasNext())$ **do**
3: $DET \leftarrow APIGetStub.GetState(DataId)$
4: $DETpart.DataId \leftarrow DET.DataId$
5: $DETpart.Holder \leftarrow DET.Holder$
6: $DETpart.ID \leftarrow DET.ID$
7: $DETpart.Sign \leftarrow DET.Sign$
8: $DETpart.DataSummary \leftarrow DET.DataSummary$
9: $DETpart.Size \leftarrow DET.Size$
10: $AllDETpart \leftarrow append(AllDETpart, DETpart)$
11: **end for**
12: **return** $All\ DETpart$

5.2.3. AccessData

DV accesses data according to *DataID* and the owned-attribute token using the data access contract. Algorithm 3 shows the specific process. The first 1–3 lines check whether the contract input is reasonable. Lines 4–14 determine whether the attribute token owned by DV meets the attribute threshold set by DO. The contract is terminated if the condition is unment, and access to the DET is prohibited. If the attribute threshold is met, lines 15–17 return the DET and decryption token of the data to the DV.

Algorithm 3 AccessData()

Input: $(DataId, \{SK_{i,j,VID}\})$
Output: $DET, \{SK_{i,VID}\}$
1: **if** $DET == null \parallel \{SK_{i,VID}\} == null$ **then**
2: **return** Error("args error")
3: **end if**
4: **for** $i = 0 \rightarrow m - 1$ **do**
5: **for** $j = 0 \rightarrow n - 1$ **do**
6: **if** $SK[i][j]! = null$ **then**
7: $count[i] + +$
8: **end if**
9: **end for**
10: **if** $count[i] < A[i].threshold$ **then**
11: **return** -1 //attribute threshold is not met, the contract is terminated
12: **end if**
13: $\{SK_{i,VID}\} \leftarrow SK_{\{i\}}$
14: **end for**
15: $DETjson \leftarrow APIGetStub.State(DataId)$
16: $DET \leftarrow json.Unmarshal(DETjson)$
17: **return** $\{SK_{i,VID}\}, DET$

6. Experiments and Results

6.1. Security Analysis

Data integrity: A semi-trusted third-party cloud storage service provider may suffer data loss due to a single failure. The distributed storage system IPFS is the off-chain data storage platform replacing cloud storage. IPFS will return the hash address based on the data content. The related hash address will change if there is any exception to the stored data. In addition, DO hashes the data and uploads them to the blockchain as a standard for integrity verification.

Privacy: The data ciphertext is uploaded to IPFS, and users without symmetric keys cannot access the data plaintext. In addition, the attribute encryption algorithm is used to encrypt the key further. The ciphertext of key in the Fabric blockchain can be viewed only when the user satisfies the attribute condition. Otherwise, only the information that does not leak data privacy can be viewed, such as data summary, size, etc.

Auditability and non-repudiation: Blockchain is used as the underlying interactive platform in this scheme. Each operation in the data sharing process is recorded in the ledger as a transaction and cannot be changed. Every transaction in the blockchain needs to attach its signature. When things go wrong in the data-sharing process, they can be audited by verifying the ledger records and signatures.

Witch Attack: The Fabric consortium blockchain network is used in this scheme. Each user needs to pass the identity authentication by Fabric-CA and can join the network after obtaining identity certificates. This not only effectively prevents identity forgery, but also prevents a node from applying for multiple identities.

6.2. Scheme Analysis

The comparison of some functions between this scheme and other blockchain-based data sharing schemes is listed in Table 2. In reference [21], data access control is realized using an elliptic curve digital signature and encryption algorithm. However, this scheme requires multiple transactions between data owners and users to complete the signature and authentication, which is inefficient and cannot provide fine-grained access control. Reference [23] adopted attribute-based access control, which uses blockchain to store corresponding data. However, this scheme has the security risk caused by semi-trusted third-party centralized authorization. In reference [31], the multi-attribute structure is used to achieve decentralized access control. However, the problems of data tampering and disclosure caused by semi-trusted cloud storage services are not considered. Therefore, the distributed storage platform IPFS is used as an off-chain storage tool to eliminate data duplication and solve storage problems. Using encryption technology completes the data access control. Moreover, the smart contract is applied to achieve data upload, query and access. Blockchain records every process in the system and provides an auditable ledger.

Table 2. Scheme function comparison.

Scheme	Distributed Data Storage	Data Integrity	Eliminate Duplicate Data	ABAC	Decentralized Access Authorization
[21]	Y	N	Y	N	N
[23]	Y	Y	N	Y	N
[31]	N	N	N	Y	Y
Ours	Y	Y	Y	Y	Y

Y: This means that the scheme proposed in the literature has this function. N: This means that the scheme proposed in the literature does not have this function.

6.3. System Performance Analysis

We conduct experiments using the proposed scheme to evaluate its performance in all directions and compare it with other schemes. The overall experimental environment includes Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz client. The Fabric blockchain network and IPFS network are deployed in Ubuntu 21.10 LTS, and the chaincode used in the scheme is developed with the Golang language. In this experiment, Hyperledger-TWGC/Tape testing tool is used to evaluate the blockchain. Tape is a lightweight tool for testing Fabric performance, allowing users to customize test conditions to detect transaction latency, throughput, and more in the blockchain.

6.3.1. Data Storage Analysis

This experiment compares the upload and download times of files with different sizes in distributed IPFS networks and traditional centralized cloud storage. Experiments are

conducted, including seven groups of different files sizes, namely, 10 MB, 50 MB, 100 MB, 200 MB, 300 MB, 400 MB and 500 MB. The upload and download time of different size files are shown in Figures 9 and 10. The figure clearly shows that the time required for uploading and downloading will increase with the increase in the file size. However, there is no obvious gap between the IPFS network and traditional cloud storage. If IPFS is used to store data, it can effectively solve the security problems that may be caused by third-party cloud storage.

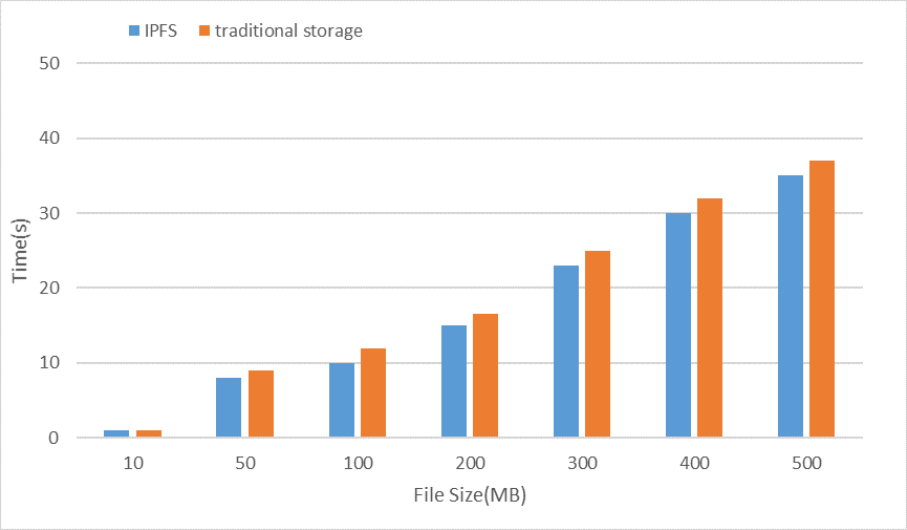

Figure 9. Time consumption comparison of uploading different size files using IPFS network and traditional cloud storage.

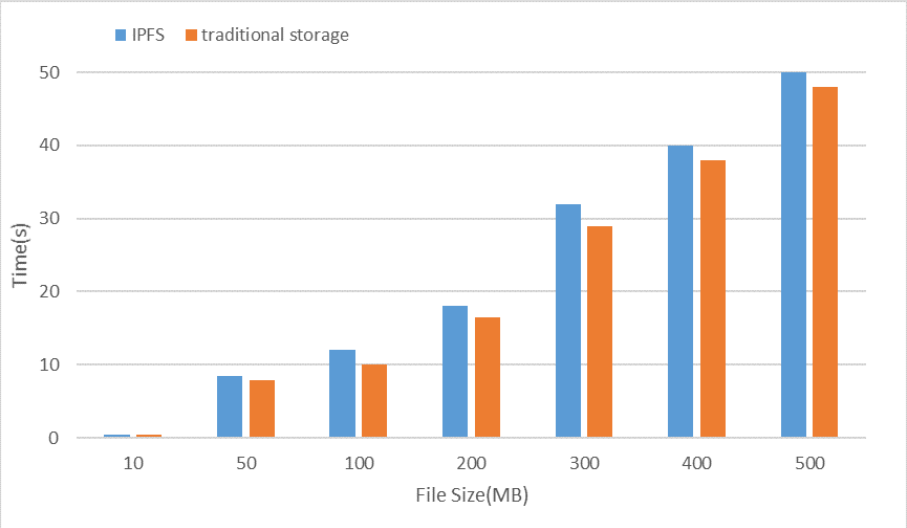

Figure 10. Time consumption comparison of downloading different size files using IPFS network and traditional cloud storage.

6.3.2. Smart Contract

This experiment tested the contract throughput of three smart contracts deployed in blockchain networks under different total amounts of transaction. Throughput (TPS) is the speed at which the blockchain ledger receives transactions, measured by the number of transactions executed per second. Figure 11 shows the contract TPS of **UploadData** under different transaction amounts in the blockchain. A comparative experiment was also performed with a different number of attributes (5, 10, and 15) involved in constructing access control policies. Figure 12 shows the TPS of the **AccessData** under different transac-

tion numbers when the attribute number is 10 or 20, respectively. The figure shows that when the size and transmission rate of blocks in the blockchain remain unchanged, the TPS increases slightly with the increase in the transaction amount and finally tends to a maximum value. As the number of attributes increases, the TPS of the contract decreases, but the maximum reduction will not exceed 50.

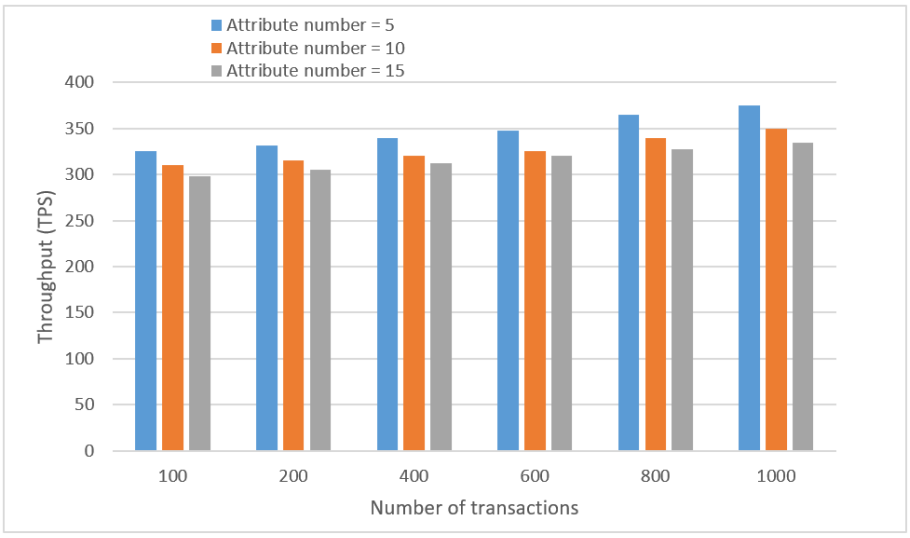

Figure 11. Throughput of **UploadData** under different transaction numbers.

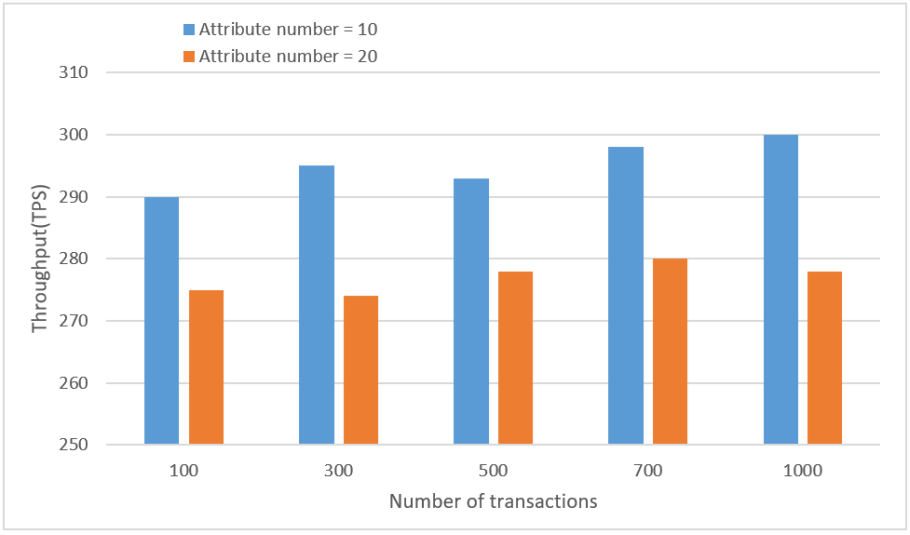

Figure 12. Throughput of **AccessData** under different transaction numbers.

Figure 13 shows the TPS of the **ReadData** under different transaction numbers when the number of existing DET in the current blockchain ledger is 10 and 20, respectively. The figure shows that when the size and transmission rate of blocks in the blockchain remains unchanged, the TPS remains basically unchanged as the transaction amount increases. However, as the total number of DET increases, the throughput of the system decreases.

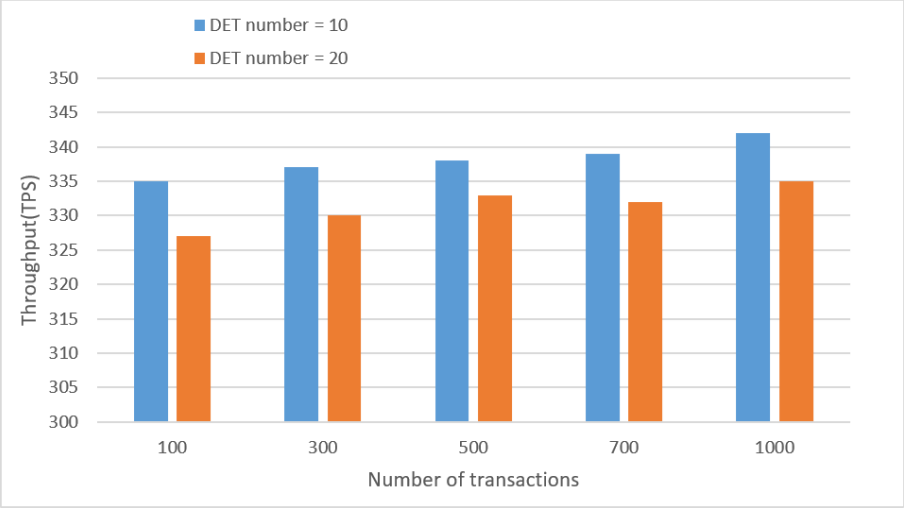

Figure 13. Throughput of **ReadData** under different transaction numbers.

6.3.3. Attribute Encryption Analysis

The ECC-improved MA-CPABE access control algorithm used in the proposed scheme is analyzed and compared to the BMAC scheme proposed in [31]. The BMAC scheme uses the traditional MA-CPABE scheme. The difference between the two encryption algorithms is considered in the experiment. In addition to the four algorithms other than system initialization, the relationship between their time cost and the number of attributes is shown in Figures 14–17. The figures show that the improved attribute encryption algorithm using ECC in the proposed scheme is superior to the traditional MA-CPABE at all stages. The simple scalar multiplication on the ECC is used to replace the traditional complex bilinear pairing in this paper, significantly reducing the computation overhead.

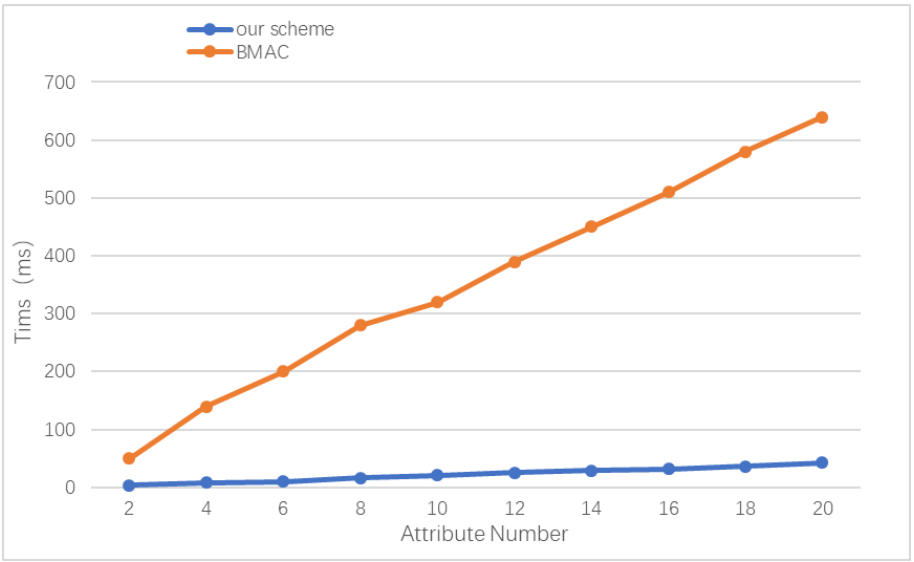

Figure 14. Time overhead of algorithm $Authority\_Setup\{(PP \rightarrow (PK, SK)\}$ under different attribute number.

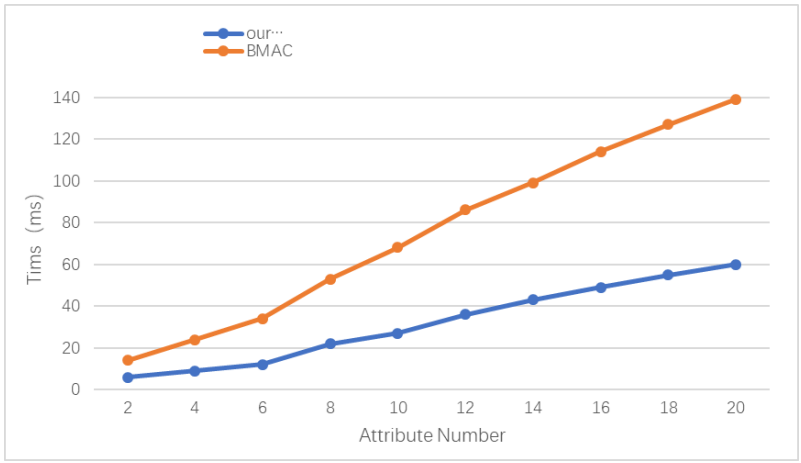

Figure 15. Time overhead of algorithm $KeyGen\{(PP, i, SK, GID) \rightarrow SK_{i,GID}\}$ under different attribute number.

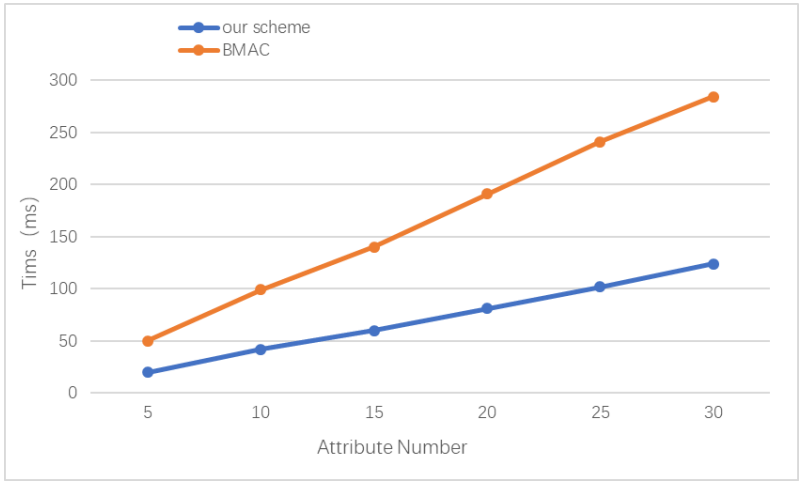

Figure 16. Time overhead of algorithm $Encrypt\{(PP, (A, \rho), M, \{PK\}) \rightarrow CT\}$ under different attribute number.

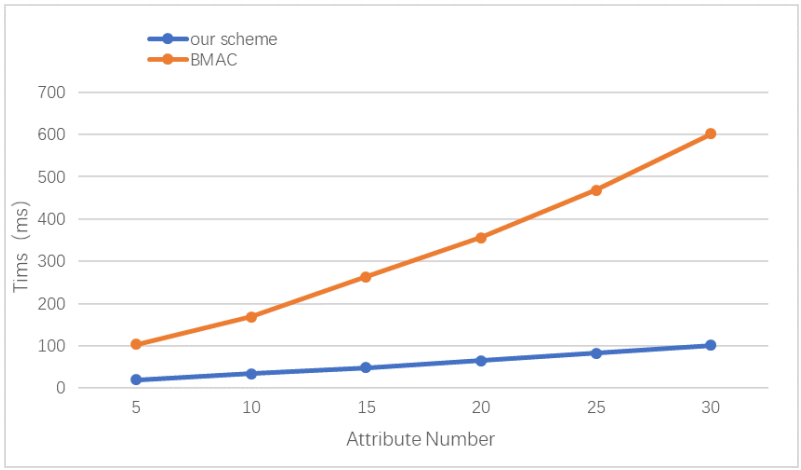

Figure 17. Time overhead of algorithm $Decrypt\{(PP, CT, SK_{i,GID}) \rightarrow M\}$ under different attribute number.

7. Conclusions

Although the traditional centralized approach has brought some convenience to data sharing, the management of data access rights is completely controlled by the central

authority organization, which will lead to trust problems. Additionally, because the authority management is not transparent, the security of the data sharing process cannot be guaranteed. In order to solve some problems relating to traditional data sharing schemes, this paper proposes a multi-authority attribute access control scheme based on Fabric blockchain. First, IPFS is used as an off-chain storage container in this solution to solve the problems of data tampering, privacy disclosure and redundant storage in centralized cloud storage. Then, distributed access control is implemented in Fabric through ECC's improved multi-permission attribute encryption algorithm and attribute threshold. It can protect data more safely and reduce the consumption of computing resources by users. The three smart contracts designed in the scheme are used to achieve data upload, query and secure access. Because the Fabric blockchain can provide auditable operation logs, it can effectively solve the trust problem and make the whole data sharing process more transparent. Finally, the system scheme is evaluated through a large number of experiments. The results show that compared with other data sharing schemes, the scheme shows a great improvement in performance, security and practicality.

In the future, we will further study how to implement efficient and secure access control policies and dynamic update mechanisms of attribute authentication in blockchain-based data sharing schemes.

Author Contributions: Conceptualization, B.X. and Y.-P.Z.; methodology, B.X., Y.-P.Z., C.-Y.W. and X.-Y.Y.; validation, B.X. and Y.-P.Z.; data analysis, B.X.; investigation, B.X. and Y.-P.Z.; writing—original draft preparation, B.X.; writing—review and editing, B.X. and Y.-P.Z.; supervision, Y.-P.Z.; funding acquisition, Y.-P.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported, in part, by The Undergraduate Education Teaching Reform Project of Fujian Province of China (No. FBJG20220128), The National Social Science Fund of China (No. 21XTQ015) and 2022 Undergraduate Innovation and Entrepreneurship Training Program Project (No. 202210402011).

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qiu, J.; Tian, Z.; Du, C.; Zuo, Q.; Su, S.; Fang, B. A survey on access control in the age of internet of things. *IEEE Internet Things J.* **2020**, *7*, 4682–4696. [CrossRef]
2. Feng, C.; Yu, K.; Bashir, A.K.; Al-Otaibi, Y.D.; Lu, Y.; Chen, S.; Zhang, D. Efficient and secure data sharing for 5G flying drones: A blockchain-enabled approach. *IEEE Netw.* **2021**, *35*, 130–137. [CrossRef]
3. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
4. Nakamoto, S.; Bitcoin, A. A peer-to-peer electronic cash system. *Bitcoin* **2008**, *4*. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 1 February 2023).
5. Javaid, M.; Haleem, A.; Singh, R.P.; Khan, S.; Suman, R. Blockchain technology applications for Industry 4.0: A literature-based review. *Blockchain Res. Appl.* **2021**, *2*, 100027. [CrossRef]
6. Sayeed, S.; Marco-Gisbert, H. Assessing blockchain consensus and security mechanisms against the 51% attack. *Appl. Sci.* **2019**, *9*, 1788. [CrossRef]
7. Sayeed, S.; Marco-Gisbert, H.; Caira, T. Smart contract: Attacks and protections. *IEEE Access* **2020**, *8*, 24416–24427. [CrossRef]
8. Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.
9. Sun, P. Security and privacy protection in cloud computing: Discussions and challenges. *J. Netw. Comput. Appl.* **2020**, *160*, 102642. [CrossRef]
10. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-policy attribute-based encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
11. Lewko, A.; Waters, B. Decentralizing attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011: Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15–19 May 2011*; Proceedings 30; Springer: Berlin/Heidelberg, Germany, 2011, pp. 568–588.
12. Rouselakis, Y.; Waters, B. Efficient statically-secure large-universe multi-authority attribute-based encryption. In *Financial Cryptography and Data Security: Proceedings of the 19th International Conference, FC 2015, San Juan, Puerto Rico, 26–30 January 2015*; Revised Selected Papers; Springer: Berlin/Heidelberg, Germany, 2015; pp. 315–332.

13. Sandhia, G.; Raja, S.K. Secure sharing of data in cloud using MA-CPABE with elliptic curve cryptography. *J. Ambient. Intell. Humaniz. Comput.* **2022**, *13*, 3893–3902. [CrossRef]
14. Wang, Q.; Jin, H. Data leakage mitigation for discretionary access control in collaboration clouds. In Proceedings of the 16th ACM Symposium on Access Control Models and Technologies, Innsbruck Austria, 15–17 June 2011; pp. 103–112.
15. Kamboj, P.; Khare, S.; Pal, S. User authentication using Blockchain based smart contract in role-based access control. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 2961–2976. [CrossRef]
16. Hu, V.C.; Kuhn, D.R.; Ferraiolo, D.F.; Voas, J. Attribute-based access control. *Computer* **2015**, *48*, 85–88. [CrossRef]
17. Chase, M. Multi-authority attribute based encryption. In *Theory of Cryptography: Proceedings of the 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, 21–24 February 2007*; Proceedings 4; Springer: Berlin/Heidelberg, Germany, 2007; pp. 515–534.
18. Yang, J.; Wen, J.; Jiang, B.; Wang, H. Blockchain-based sharing and tamper-proof framework of big data networking. *IEEE Netw.* **2020**, *34*, 62–67. [CrossRef]
19. Guo, S.; Hu, X.; Guo, S.; Qiu, X.; Qi, F. Blockchain meets edge computing: A distributed and trusted authentication system. *IEEE Trans. Ind. Inform.* **2019**, *16*, 1972–1983. [CrossRef]
20. Alshalali, T.; M'Bale, K.; Josyula, D. Security and privacy of electronic health records sharing using hyperledger fabric. In Proceedings of the 2018 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 12–14 December 2018; pp. 760–763.
21. Chen, C.L.; Yang, J.; Tsaur, W.J.; Weng, W.; Wu, C.M.; Wei, X. Enterprise data sharing with privacy-preserved based on hyperledger fabric blockchain in IIOT's application. *Sensors* **2022**, *22*, 1146. [CrossRef]
22. Liu, H.; Han, D.; Li, D. Fabric-IoT: A blockchain-based access control system in IoT. *IEEE Access* **2020**, *8*, 18207–18218. [CrossRef]
23. Lu, X.; Fu, S.; Jiang, C.; Lio, P. A fine-grained IoT data access control scheme combining attribute-based encryption and blockchain. *Secur. Commun. Netw.* **2021**, *2021*, 5308206. [CrossRef]
24. Liang, W.; Yang, Y.; Yang, C.; Hu, Y.; Xie, S.; Li, K.C.; Cao, J. PDPChain: A consortium blockchain-based privacy protection scheme for personal data. *IEEE Trans. Reliab.* **2022**, 1–13. [CrossRef]
25. Eltayieb, N.; Elhabob, R.; Hassan, A.; Li, F. A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud. *J. Syst. Archit.* **2020**, *102*, 101653. [CrossRef]
26. Cong, R.; Liu, Y.; Tago, K.; Li, R.; Asaeda, H.; Jin, Q. Individual-initiated auditable access control for privacy-preserved iot data sharing with blockchain. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
27. Gao, H.; Ma, Z.; Luo, S.; Xu, Y.; Wu, Z. BSSPD: A blockchain-based security sharing scheme for personal data with fine-grained access control. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6658920. [CrossRef]
28. Zhang, L.; Kan, H.; Huang, H. Patient-centered cross-enterprise document sharing and dynamic consent framework using consortium blockchain and ciphertext-policy attribute-based encryption. In Proceedings of the 19th ACM International Conference on Computing Frontiers, Turin, Italy, 17–22 May 2022; pp. 58–66.
29. Guo, H.; Li, W.; Nejad, M.; Shen, C.C. Access control for electronic health records with hybrid blockchain-edge architecture. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; pp. 44–51.
30. Sammy, F.; Vigila, S. An efficient blockchain based data access with modified hierarchical attribute access structure with CP-ABE using ECC scheme for patient health record. *Secur. Commun. Netw.* **2022**, *2022*, 8685273. [CrossRef]
31. Qin, X.; Huang, Y.; Yang, Z.; Li, X. A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing. *J. Syst. Archit.* **2021**, *112*, 101854. [CrossRef]
32. Hankerson, D.; Menezes, A.J.; Vanstone, S. *Guide to Elliptic Curve Cryptography*; Springer Science Business Media: New York, NY, USA, 2006.
33. Beimel, A. *Secure Schemes for Secret Sharing and Key Distribution*; Technion-Israel Institute of Technology & Faculty of Computer Science: Haifa, Israel, 1996.
34. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto Portugal, 23–26 April 2018; pp. 1–15.
35. Trautwein, D.; Raman, A.; Tyson, G.; Castro, I.; Scott, W.; Schubotz, M.; Gipp, B.; Psaras, Y. Design and evaluation of IPFS: A storage layer for the decentralized web. In Proceedings of the ACM SIGCOMM 2022 Conference, Amsterdam, The Netherlands, 22–26 August 2022; pp. 739–752.