Research Article

A Fine-Grained IoT Data Access Control Scheme Combining Attribute-Based Encryption and Blockchain

Xiaofeng Lu,[1] **Songbing Fu**,[1] **Cheng Jiang**,[1] **and Pietro Lio**[2]

[1]*Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*Computer Laboratory, University of Cambridge, Cambridge, UK*

Correspondence should be addressed to Xiaofeng Lu; luxf@bupt.edu.cn

Received 11 June 2021; Revised 30 July 2021; Accepted 23 August 2021; Published 16 September 2021

Academic Editor: AnMin Fu

IoT technology has been widely valued and applied, and the resulting massive IoT data brings many challenges to the traditional centralized data management, such as performance, privacy, and security challenges. This paper proposes an IoT data access control scheme that combines attribute-based encryption (ABE) and blockchain technology. Symmetric encryption and ABE algorithms are utilized to realize fine-grained access control and ensure the security and openness of IoT data. Moreover, blockchain technology is combined with distributed storage to solve the storage bottleneck of blockchain systems. Only the hash values of the data, the hash values of the ciphertext location, the access control policy, and other important information are stored on the blockchain. In this scheme, smart contract is used to implement access control. The results of experiments demonstrate that the proposed scheme can effectively protect the security and privacy of IoT data and realize the secure sharing of data.

1. Introduction

At present, the Internet of Things (IoT) technology has been more and more widely used [1], such as in smart medicine and smart cars. IoT devices generate data in real-time and share data by connecting to the Internet. In most traditional IoT systems, data storage and computing adopt a centralized architecture, but the massive, heterogeneous data characteristics and multidimensional, real-time service requests pose great challenges to the centralized data management architecture [2]. Centralized storage and computing architectures are vulnerable to various network attacks, such as data tampering attack, single-point attacks, and distributed denial-of-service attacks [3]. IoT data is a true description of the physical world and often involves personal privacy. If these data are maliciously tampered, forged, or illegally accessed, it may cause disastrous consequences.

A blockchain is a decentralized, tamper-proof, traceable, and multiparty distributed database that integrates a P2P protocol, asymmetric encryption technology, a consensus mechanism, and other technologies [4]. The data on a blockchain are transparent to each participant, and as long as an honest node holds most of the CPU computing power, the system is safe and reliable. Therefore, the security mechanism of blockchain can effectively solve the problem of data authenticity in the application of IoT, and to a certain extent, it can guarantee the confidentiality and availability of industrial IoT data [5]. Smart contracts—self-executing scripts that reside on the blockchain—allow for distributed, heavily automated workflows. This should make blockchains very suitable for combination with IoT applications [6]. Ge et al. [7] combined blockchain technology with IoT and proposed a decentralized security mechanism based on blockchain technology; the key data generated by IoT devices are stored in the blockchain, and the privacy and security problems that may be encountered by the centralized IoT are solved. Li et al. [8] applied blockchain technology to the medical field and proposed a data protection system (DPS) architecture that stores important medical records on the blockchain to achieve the purposes of data being tamper-proof and traceable.

On the contrary, a centralized storage architecture results in data producers having no control over the data. For instance, large websites collect not only personal

information, such as users' hobbies and browsing habits, but also personal privacy data, which introduces potential threats to personal privacy security. Therefore, it is necessary to strengthen the data owner's control over the data and decide whether other users have the right to access the data [9, 10]. In 2005, attribute-based encryption (ABE) was first proposed by Waters to support the integration of data privacy protection and access control [11]. In the ABE mechanism, the user's private key and ciphertext are associated with a group of attributes, and the user can successfully decrypt the plaintext only when the number of attribute intersections between the attribute set associated with the user's private key and that associated with the ciphertext reaches the threshold value set by the system.

However, there remain many defects and vulnerabilities in blockchain technology, among which storage is one of the most notable [12]. For instance, the Bitcoin system produced a total of 546,349 blocks by October 18, 2018, and the size of a block is generally 996.2 kB, and the size of the entire blockchain is 186.9 GB [13]. With the passage of time, the storage space required by blockchain systems will become increasingly larger. If all the data generated by IoT devices are stored on the blockchain, the size of the entire blockchain will be very large. However, IoT devices cannot store such a large blockchain, so storage optimization is an important challenge that must be overcome. The interplanetary file system (IPFS) is a point-to-point distributed file system [14] that aims to connect all computing devices with the same file system and can be used as a storage scheme for blockchains. Xu et al. [15] proposed a decentralized social network system based on Ethereum and IPFS, which uses IPFS to save large amounts of file data to reduce the Ethereum storage pressure. Klems et al. [16] proposed a decentralized service market system based on blockchain technology and IPFS. In this system, large amounts of data and service metadata are stored in the IPFS network, while only the hash values of these data are stored in the blockchain.

This paper proposes an IoT data access control scheme that combines an ABE algorithm and blockchain technology. In view of the privacy and security problems of IoT data, a symmetric encryption algorithm is combined with an ABE algorithm. Only users who meet the access control policy can successfully access the data and achieve fine-grained access control. To solve the problem of the low storage capacity of blockchain network nodes, the "off-chain" storage mode is adopted. The data are encrypted by a symmetric encryption algorithm, and the ciphertext of the data is stored in the IPFS network. The address hash value of encrypted data stored on IPFS is saved on the blockchain. In this way, the blockchain network is associated with IPFS, which reduces the burden of blockchain storage. Moreover, a consortium blockchain system was constructed based on the Hyperledger Fabric framework, and experiments were carried out to prove the feasibility of the scheme.

At present, there have been many access control schemes based on ciphertext policy attribute-based encryption (CP-ABE) [17] and blockchain, but these schemes often focus on the granularity of access control, but pay less attention to the efficiency of data sharing. Some researchers realize that blockchain cannot store a large amount of data and adopt cloud storage, but the efficiency of cloud storage is low. The access control scheme proposed in this paper not only uses CP-ABE and blockchain to realize access control but also ensures the simplicity of the control process. Therefore, the scheme realizes the balance between fine-grained access control and efficient file sharing.

The main contributions of our work are as follows:

(1) The scheme proposed in this paper can ensure the secure and efficient sharing of IoT data, provide fine-grained access to data, simple control process, and enhance the scalability of blockchain system storage. The scheme realizes the balance between fine-grained access control and efficient file sharing.

(2) The blockchain is used to store the address hash value of a file on IPFS, data hash value, access control policy, timestamp, and other information This scheme not only ensures that the data cannot be tampered with but also ensures that the access control strategy cannot be tampered with.

2. Related Work

There have been many studies on the application of blockchain technology to data sharing and access control of IoT. For example, Ge et al. and Li et al. [8] used blockchain technology to store important data generated by IoT devices, solve the privacy and security problems of centralized storage architectures, and realize the secure sharing of data. However, these strategies are weak in terms of data access control and do not optimize the blockchain storage; thus, they easily encounter the blockchain storage bottleneck.

2.1. Access Control and Blockchain. Access control technology is widely used in all types of information systems to control the access rights of users and avoid illegal access to data. However, the traditional access control scheme mainly relies on a third-party trusted server, which is characterized by problems such as a single point of failure and low efficiency.

Many researchers have proposed attribute-based data access control methods [9, 10]. In order to ensure the confidentiality of data, an application level fine-grained data access is designed by using the attribute-based encryption method [9]. Fan et al. proposed a fine-grained access-control scheme based on CP-ABE and Trusted Execution Environment (TEE) [10]. TEE is employed as a trusted computing environment to protect encrypted data [9, 10]. This paper uses blockchain technology to ensure the integrity of encrypted data. In the work by Jemel and Serhrouchni [18], a dynamic access control strategy based on blockchain technology and ciphertext policy attribute-based encryption (CP-ABE) was proposed. The time attribute is introduced to realize the dynamic access of data, and only a user whose attribute meets the access control policy within the specified time can access the data.

Blockchain technology is a new technology that can effectively solve the problems existing in traditional access

control. Many researchers have explored the combination of a blockchain and IoT access control technology. For instance, Novo [19] proposed an extensible access management architecture for IoT based on blockchain technology, which defines and executes access control rules via smart contracts. However, this scheme is based on a private blockchain network and sacrifices decentralization to improve performance.

Li et al. [20] proposed an authentication and security scheme for IoT based on blockchain technology, in which the hash values of important data are stored in the blockchain and the unique ID of the device is recorded for authentication, which can effectively avoid single-point-of-failure attacks on the certificate authority (CA) server. However, this scheme ignores the storage bottleneck of blockchains. Ding et al. [21] proposed an attribute-based access control framework, which uses a set of attributes to describe devices and records the distribution of attributes on the chain to avoid a single point of failure and data tampering. Moreover, this scheme uses signature technology and hash operations to simplify the access control protocol.

To a certain extent, while the methods proposed in these studies can achieve data access control, even fine-grained access control, there remain blockchain storage bottlenecks.

2.2. Blockchain and Storage Optimization. Compared with the traditional centralized storage architecture, blockchain technology is characterized by decentralization and non-tamperability, which can effectively guarantee the security of data and improve the scalability of the system. However, the application of blockchain technology to a storage system also introduces new challenges, such as the increase of the storage space overhead.

To solve the blockchain storage problem, many investigations have been carried out. In the work by Zhang and Wang [12], a blockchain fragmentation storage model based on threshold secret sharing was proposed. By improving the Shamir threshold, instead of storing complete transaction data in each node, the data on the chain are segmented and stored, which can effectively reduce the storage capacity of each node. Dai et al. [22] proposed a blockchain storage architecture called NC-DS, which uses network coding to encode data and save the storage space of the blockchain.

In these previous studies, data were dealt with directly. The method proposed by Zhang and Wang [12] stores the data in pieces, while the method proposed by Dai et al. [22] encodes the data. Both schemes can reduce the storage cost of blockchains, but they are both improved on the basis of storing a complete ledger, which saves little space and weakens access control. Cheng et al. [23] proposed a data management scheme for IoT based on blockchain technology and edge computing, which uses the Advanced Encryption Standard (AES) encryption algorithm to protect data security and personal privacy, stores the hash values and some important files on the chain, and stores the encrypted data on the edge server by using a distributed algorithm (Kademlia) to solve the storage bottleneck problem of the blockchain system.

Some data sharing schemes combine the CP-ABE algorithm with blockchain and cloud storage. Wang and Song [24] used attribute-based encryption (ABE) and identity-based encryption (IBE) to encrypt medical data and used identity-based signature (IBS) to implement digital signatures. Wang et al. proposed a personal health records sharing scheme based on blockchain [25]. Wang's scheme used searchable symmetric encryption and attribute-based encryption techniques to achieve fine-grained access control. Wang's scheme allowed patients to distribute attribute private key for users. Both [24, 25] uses the cloud to store medical data, but the security of data privacy depends on cloud service providers.

However, in each data access process, data consumers must communicate with data owners to obtain access rights, which takes a considerable amount of time. In the scheme proposed in this paper, the encrypted data are stored in an IPFS distributed network, and the CP-ABE algorithm is used to achieve fine-grained data access control. The blockchain only stores the hash values of data, the content hash values generated by IPFS, the access control policy, timestamps, and other metadata information, which greatly reduces the storage overhead.

3. Data Access Control Scheme

3.1. Scheme Architecture. The architecture of the data access control scheme that combines ABE and blockchain technology consists of five layers, namely, the consumption layer, interaction layer, access control layer, data layer, and IoT devices layer, as shown in Figure 1. The consumption layer contains all kinds of data consumers, software, or hardware. The interaction layer provides good access protocols and services for data consumers. The access control layer is the concrete realization of business logic, including the realization of smart contracts and the data control access algorithm. Finally, the data layer consists of the blockchain network and IPFS distributed network. The bottom layer is IoT devices layer. IoT devices generate all kinds of data.

The blockchain network is used to store the hash values of data, the content hash values generated by IPFS, the access control policy, timestamps, and other information. Data consumers must meet the stipulations of the access control policy on the blockchain to access the data. After successful access, the consistency and integrity of the data can also be verified through the blockchain. The IPFS distributed network is composed of several server devices with good performance. The data generated by IoT devices will be encrypted and stored in IPFS, and the hash content values generated by IPFS will be stored on the blockchain. In this way, the blockchain network is associated with IPFS, thereby reducing the burden of blockchain storage. The third-party authorization server mainly generates and transmits the public key (PK) and master key (MK) generated by the CP-ABE initialization algorithm and the private key (SK) generated by the CP-ABE key generation algorithm. In this architecture, users do not need to join the blockchain network, let alone consider the specific implementation details of the whole access control scheme. IoT devices only

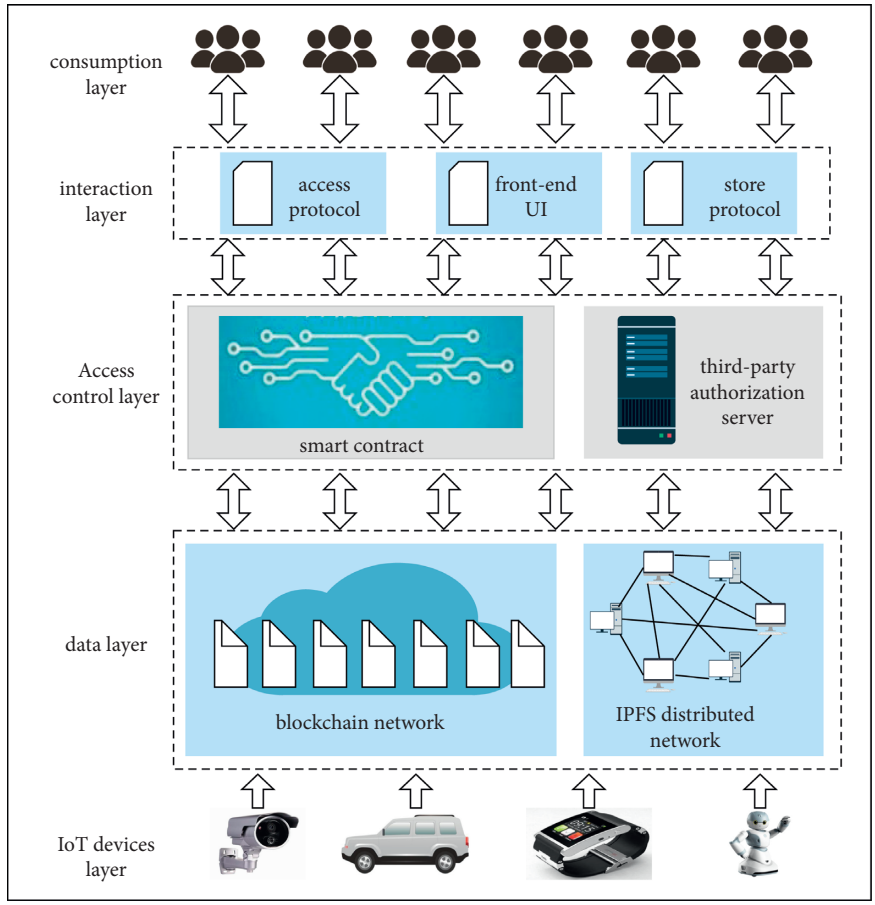

Figure 1: Architectural diagram of the scheme.

produce data or files, and data consumers only need to access data or files. In this scheme, the owner of the IoT device is called the data owner.

3.2. Blockchain Design. A blockchain is composed of many blocks, each of which contains a block header and block body. The structure of a blockchain is presented in Figure 2. The block header stores the version number (VersionId), the hash value of the previous block (PreBlock Hash), the Merkel root, and the timestamp (TimeStamp). PreBlock Hash links isolated blocks into chains, TimeStamp refers to the generation time of the block, and the Merkle root is the hash of multiple transaction data. The tampering of transaction data will lead to the inconsistency of the Merkle root, which can be used for transaction integrity verification. The block body primarily stores the hash values of the source data (hashfile), the hash values generated by IPFS (hashipfs), and the access control strategy (policy). Among them, hashfile is generated by the SHA256 algorithm and occupies 32 bytes, hashipfs is the hash value returned after uploading the file to the IPFS network and also occupies 32 bytes, and policy is the access control strategy of the data owner; different owners have different policies, and the upper limit is 1000 bytes.

3.3. Attribute-Based Encryption Mechanism. In the ABE mechanism, the sender uses a set of attributes W to encrypt the message, and the receiver uses a set of attributes W' to describe the identity corresponding to the private key. Only when the intersection number of W' and W exceeds the threshold value t set by the system can the message receiver decrypt the ciphertext. However, this mechanism is limited by access control structures that can only support the threshold policy.

To solve this problem, Goyal et al. [26] proposed a key policy attribute-based encryption (KP-ABE) scheme, which supports fine-grained data access control. In this scheme, the ciphertext is associated with the attribute set of the system, and the key is associated with the access control structure. Only when the attribute set of the user satisfies the access control policy can the decryption be performed. Additionally, Bethencourt et al. proposed CP-ABE [17], which is different from KP-ABE. The ciphertext of CP-ABE is associated with access control, and its key is associated with the attribute set. Only when the user's attribute set satisfies this access control structure can it be decrypted. Moreover, the access control authority of CP-ABE is controlled by the message sender. Therefore, the CP-ABE encryption scheme was adopted in the present work to ensure the data owner's control over the data and realize the fine-grained access control of the data. The

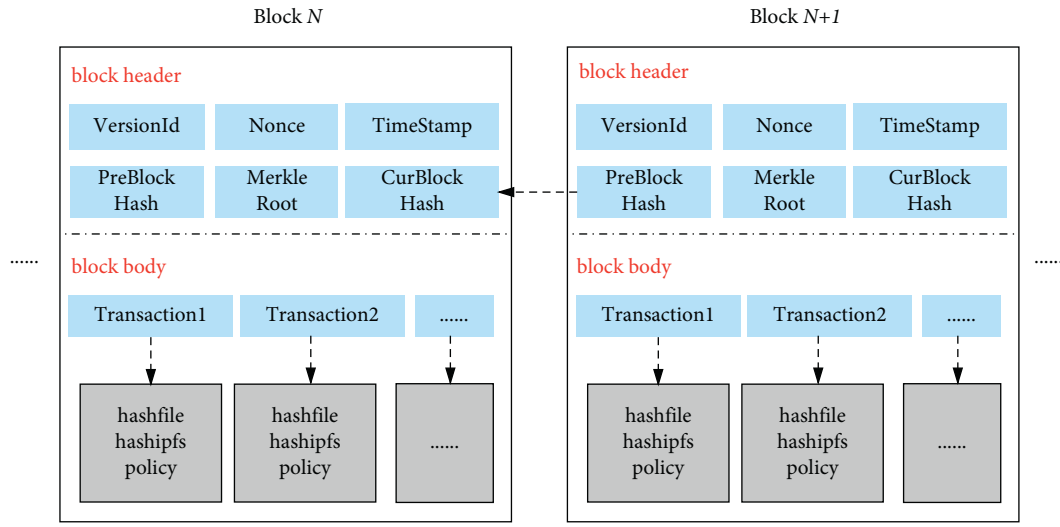

Figure 2: Blockchain structure.

algorithm flow of CP-ABE is presented in Figure 3 and is composed of four polynomial algorithms.

(1) The initialization phase: the trusted key distribution center executes the random initialization algorithm, as shown in equation (1); the input is the security parameter r, and the output includes the public key (PK) and master key (MK):

$$(PK, MK) = \text{Setup}(r). \tag{1}$$

(2) The key generation stage: the trusted key distribution center executes the key generation algorithm, as shown in equation (2); the inputs are PK and MK generated by equation (1) and the user-defined attribute set A, and the output includes a private key (SK):

$$SK = \text{KeyGen}(\text{PK}, \text{MK}, A). \tag{2}$$

(3) The data encryption stage: the data owner executes the encryption algorithm, as shown in equation (3); the inputs include PK, the message m to be encrypted, and the access structure T, and the output is the ciphertext c:

$$c = \text{Encrypt}(\text{PK}, m, T). \tag{3}$$

(4) The data decryption stage: the data requester executes the decryption algorithm, as shown in equation (4); the inputs include PK, SK, and c, and the output is the plaintext message m:

$$m = \text{Decrypt}(\text{PK}, c, \text{SK}). \tag{4}$$

Table 1 presents the description of the symbols used in this scheme.

3.4. IPFS Distributed Storage. IPFS combines the distributed hash table (DHT), incentive block exchange, self-authentication namespace, and other technologies. Moreover, the data of IPFS are distributed on different devices, and there exist multiple backups to avoid a single point of failure. Different from the existing web system, in which resources are accessed through URLs, IPFS allows for the retrieval of files by obtaining a unique hash value from the file content. Therefore, once the content of the file changes, the address of the file will change, thereby achieving tamper-proof data. With the passage of time, the storage space required by the blockchain will become increasingly larger. In the proposed method, the ciphertext of the file is stored in the IPFS network, which can alleviate the rapid expansion of the blockchain caused by too much data.

3.5. Data Storage and Access

3.5.1. Data Storage. As shown in Figure 4, the data storage procedure of this scheme includes five participants, namely, the data owner, system server side, IPFS distributed network, blockchain network, and third-party authorization server. The detailed process is as follows.

(1) The data owner (Owner) selects the file to be stored and sets the access control policy of the file (policy). The data consumer (Consumer) can successfully access the file only if the set of attributes of the data consumer meets the stipulations of the access control policy:

$$\text{policy} \longleftarrow (\text{Owner}, \text{file}). \tag{5}$$

(2) The data owner has a unique AES key (key). If the data owner has not generated the key before, the server side calls the AES key generation algorithm to generate the key, as shown in equation (6). Then, the server side calls the AES encryption algorithm to encrypt the file and obtain the encrypted file (encfile), as shown in equation (7). Finally, the server side calls the IPFS storage algorithm to store the encrypted file in the IPFS distributed network, as

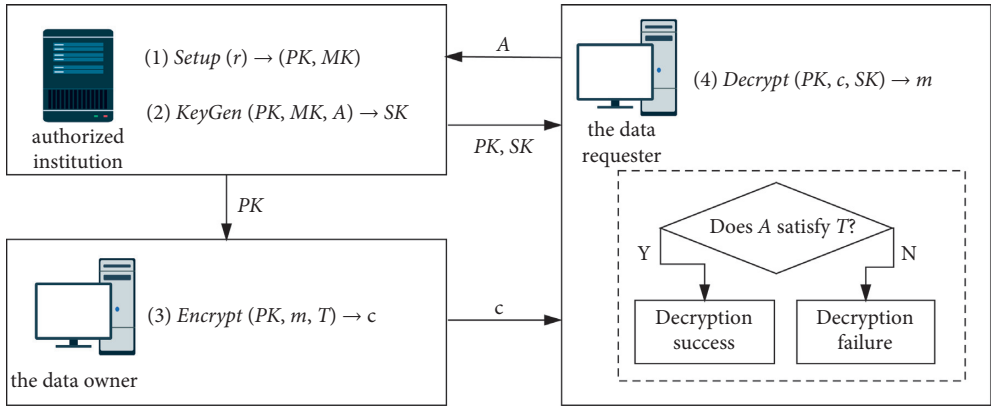

Figure 3: CP-ABE algorithm flow.

Table 1: Symbol description.

Symbol	Description
Owner	Data owner
Consumer	Data consumer
file	File
policy	Access control policy set by data owner
key	Key generated by AES algorithm
encfile	Ciphertext of a file
hashipfs	The address of the ciphertext of the file on IPFS
hashfile	The hash value of the file
PK	Public key generated by CP-ABE algorithm
MK	Master key generated by CP-ABE algorithm
SK	Private key generated by CP-ABE algorithm
A	Attribute set of data consumers
enckey	Ciphertext of key
deckey	Decrypted key
decfile	Decrypted file
dechash	Hash value of encrypted file after decryption

shown in equation (8), and records the hash value (hashipfs) used to access the ciphertext:

$$\text{key} = \text{AES.Gen}(\text{Owner}), \tag{6}$$

$$\text{encfile} = \text{AES.Enc}(\text{key}, \text{file}), \tag{7}$$

$$\text{hashipfs} = \text{IPFS.Store}(\text{encfile}). \tag{8}$$

(3) The server side calls the SHA256 algorithm to hash the file to get the file hash value (hashfile), as shown in equation (9). Then, the previously generated hashfile, hashipfs, and policy are sent to the blockchain network:

$$\text{hashfile} = \text{SHA256.Hash}(\text{file}). \tag{9}$$

(4) The blockchain network receives the data storage request and triggers the storage smart contract (StoreCont) to store the hashfile, hashipfs, and policy on the blockchain:

$$\text{StoreCont}(\text{hashfile}, \text{hashipfs}, \text{policy}). \tag{10}$$

(5) The server side requests the public key (PK) from the third-party authorization server for the later encryption of the file.

(6) The data owner has a unique public key (PK) and a unique master key (MK). If the data owner has not generated the public key and master key before, the third-party authorization server will call the initialization algorithm (Setup) of the CP-ABE algorithm to generate and store PK and MK, as shown in equation (11), and will then send PK to the server side:

$$\text{PK}, \text{MK} = \text{CPABE.Setup}(r). \tag{11}$$

(7) The server side calls the encryption algorithm (Encrypt) of the CP-ABE algorithm, takes the policy and PK as the input of the encryption algorithm, encrypts the *key* to get the ciphertext of the *key* (enckey), and stores it, as shown in the following equation:

$$\text{enckey} = \text{CPABE.Encrypt}(\text{PK}, \text{key}, \text{policy}). \tag{12}$$

The data storage algorithm is described in Algorithm 1.

3.5.2. Data Access. As shown in Figure 5, the data access process of this scheme includes five participants, namely, the data consumer, system server side, IPFS distributed network, blockchain network, and third-party authorization server. The detailed process is as follows.

(1) The data consumer (Consumer) sends out a request to access the file, which contains the attribute set A of the data consumer.

(2) After receiving the request from the data consumer, the server side requests the hash value of the file (hashfile) and the hash value used to access the ciphertext of the file to the IPFS network (hashipfs) from the blockchain network.

(3) The blockchain network receives the data access request, triggers a query smart contract

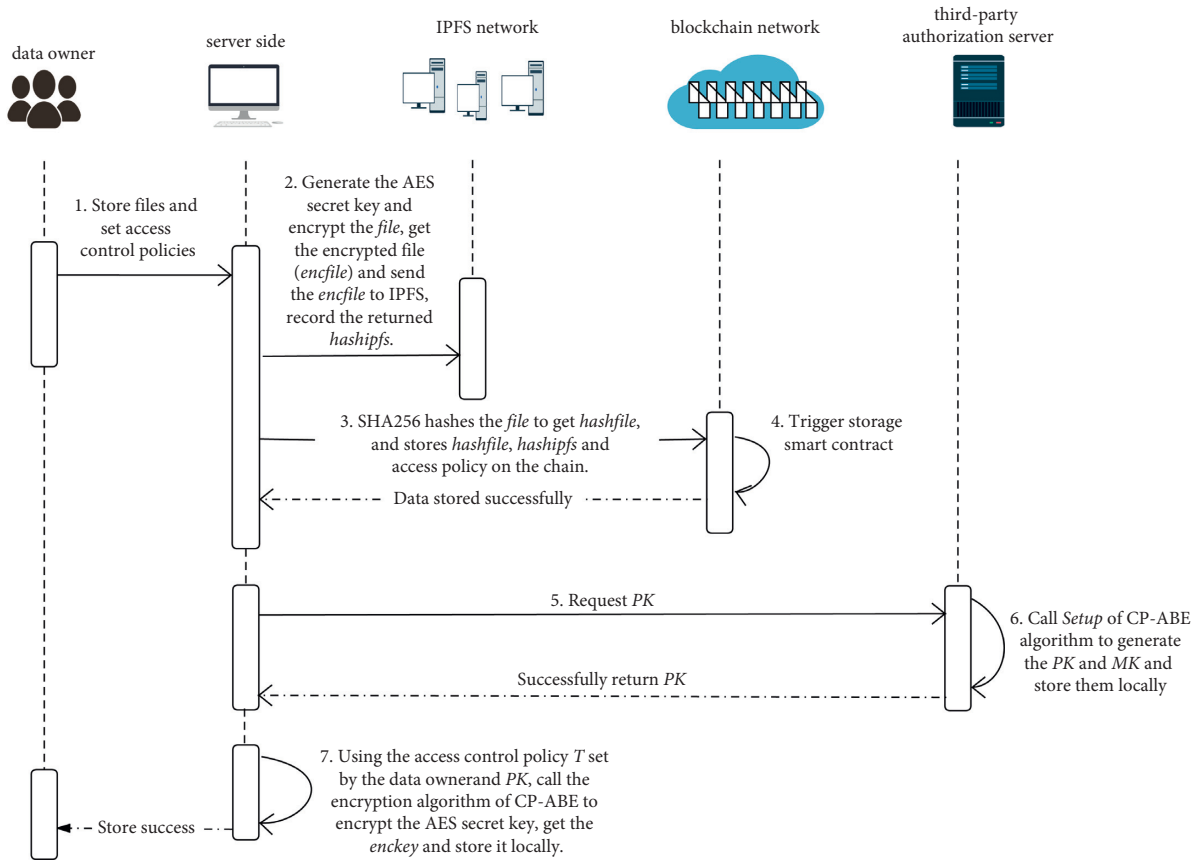

Figure 4: Data storage flow.

```
Begin
  policy ⟵ (Owner, file)
  if key not exist then
    key = AES. Gen(Owner)
  end if
  encfile = AES. Enc(key, file)
  hashipfs = IPFS. Store(encfile)
  hashfile = SHA256. Hash(file)
  StoreCont(hashfile, hashipfs, policy)
  if PK, MK not exist then
    PK, MK = CPABE. Setup(r)
  end if
  enckey = CPABE. Encrypt(PK, key, policy)
End
```

Algorithm 1: Data Storage.

(QueryCont), gets the hashfile and hashipfs, and sends them to the server side:

$$\text{hashfile, hashipfs} \longleftarrow \text{QueryCont}(\text{file}). \quad (13)$$

(4) The server side requests the public key PK and private key SK from the third-party authorization server to later decrypt the file.

(5) According to PK, MK, and attribute set A of the data consumer, the third-party authorization server executes the key generation algorithm (KeyGen) of the CP-ABE algorithm to generate the private key (SK), as shown in equation (14), and sends PK and SK to the server side:

$$\text{SK} = \text{CPABE.KeyGen}(\text{PK}, \text{MK}, \text{A}). \quad (14)$$

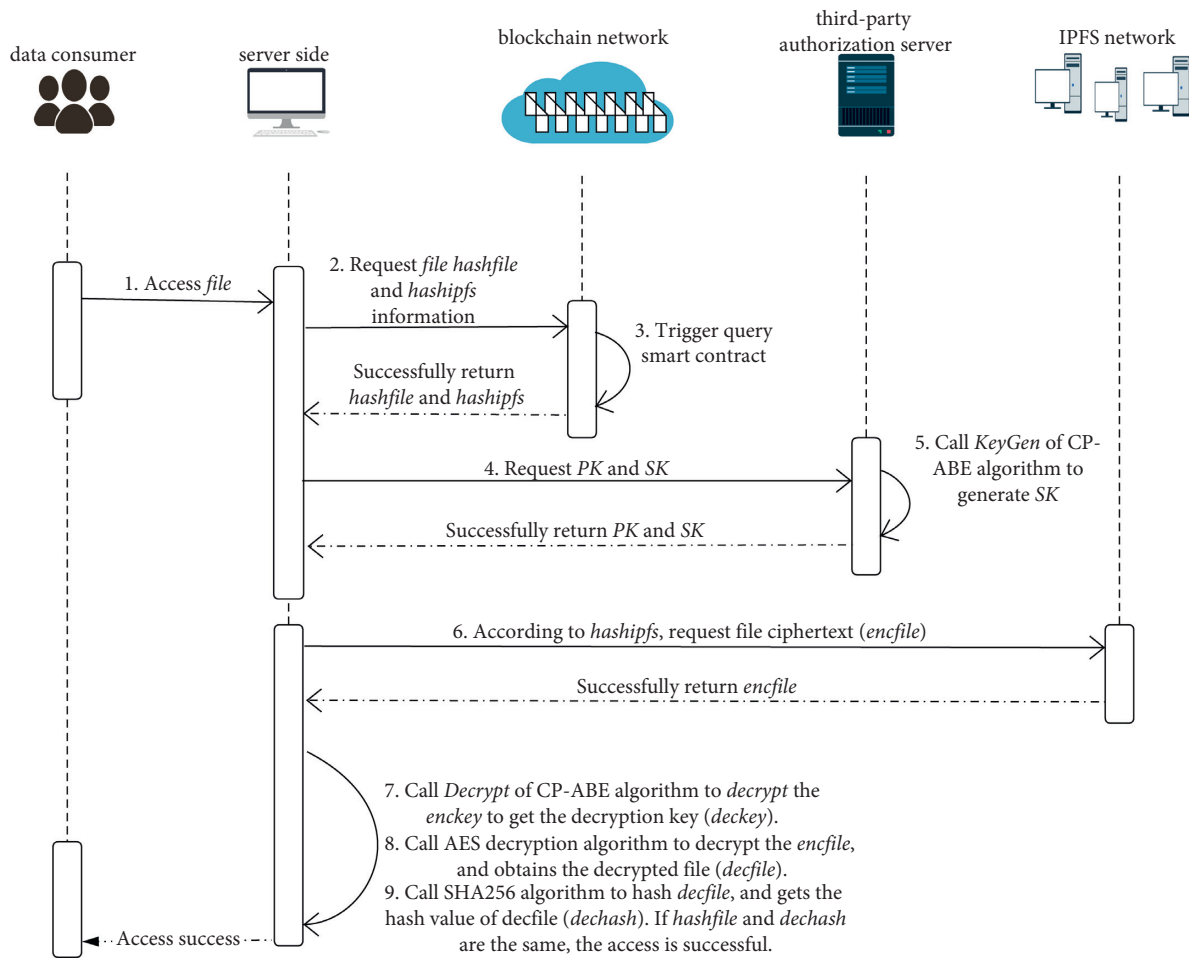

Figure 5: Data access flow.

(6) According to the hashipfs obtained from the chain, the server side calls the IPFS query algorithm to obtain the ciphertext of the file (encfile) from the IPFS network, as shown in the following equation:

$$\text{encfile} = \text{IPFS.Query}(\text{file}, \text{hashipfs}). \quad (15)$$

(7) The server side obtains the AES key ciphertext (enckey) that encrypts the file locally and calls the decryption algorithm (Decrypt) of the CP-ABE algorithm to decrypt enckey and obtain the decryption key (deckey), as shown in the following equation:

$$\begin{aligned} \text{deckey} &= \text{CPABE.Decrypt}(\text{PK}, \text{enckey}, \text{SK}), \\ \text{enckey} &= \text{Get}(\text{file}). \end{aligned} \quad (16)$$

(8) According to deckey, the server side calls the AES decryption algorithm to decrypt encfile and obtains the decrypted file (decfile), as shown in the following equation:

$$\text{decfile} = \text{AES.Dec}(\text{deckey}, \text{encfile}). \quad (17)$$

(9) The server side calls the SHA256 algorithm to hash decfile and gets the hash value of decfile (dechash), as shown in equation (18). If hashfile and dechash are the same, the access is successful:

$$\text{dechash} = \text{SHA256.Hash}(\text{decfile}). \quad (18)$$

The data access algorithm is shown in Algorithm 2.

4. Security Model

4.1. Tamper-Proof. In environmental monitoring, equipment detection, and other scenarios, a large number of IoT devices will be used. For example, a variety of sensors are used to detect the quality of large-scale equipment. Dishonest device manufacturers can make their unqualified products pass the detection by tampering the detection data, which is the threat of data tampering in the IoT.

The data block on the blockchain contains a timestamp and the hash value of the previous block. The blocks linked in the chronological order ensure that the transaction messages cannot be modified arbitrarily unless 51% of the nodes in the whole network are tampered, which is almost impossible in a large blockchain network. Therefore, this scheme can avoid the threat of data tampering.

```
Begin
  hashfile, hashipfs ⟵ QueryCont(file)
  if SK not exist then
     SK = CPABE. KeyGen (PK, MK, A)
  end if
  encfile = IPFS. Query (file, hashipfs)
  enckey = Get (Owner)
  deckey = CPABE. Decrypt (PK, enckey, SK)
  decfile = AES. Dec (deckey, encfile)
  dechash = SHA256. Hash (decfile)
  If dechash equal hashfile then
     return success
  Else
     return failure
End
```

Algorithm 2: Data access.

4.2. Confidentiality and Integrity. Some IoT applications will store users' private information. For example, the medical IoT system will store patients' condition, and the traffic monitoring system will store vehicles' video. Hackers will attack this kind of IoT applications to illegally obtain the private data, resulting in the threat of illegal access and data leakage.

The data access control scheme proposed in this paper ensures the confidentiality and integrity of system data. Via the combination of the AES symmetric encryption algorithm and CP-ABE algorithm, the data owner formulates the control access policy. Only when the attribute set owned by the data visitor meets the stipulations of the policy can the data be accessed. The data access permission of this scheme is determined by the data owner, and the fine-grained access control and sharing of data are realized by setting different access policies. During the process of data storage and access, the data are transmitted in an encrypted state, and visitors outside the data access authority cannot obtain the real data, which ensures data confidentiality. Moreover, the hash algorithm is used to hash the original data, and the hash values are stored on the blockchain. The data visitor calculates the hash value after receiving the data and compares it with the hash value on the chain; if the data has been tampered with, the two hash values will be inconsistent, which allows for the verification of the integrity of the data.

4.3. Scalability and Reliability. IoT devices usually run for a long time, and many devices work in the field or unmanned environment for a long time, such as water quality monitoring sensors. IoT data servers are prone to various failures due to long hours of work or are subject to DOS attacks, resulting in data loss or system crashes. This is called the single-point-of-failure threat.

The data access control scheme proposed in this paper ensures the reliability and robustness of the system. Traditional centralized storage and computing architectures are prone to a single point of failure. If a device is damaged, all devices may be affected, thereby causing irreparable losses. In this work, decentralized blockchain technology and IPFS distributed technology were combined to store the ciphertext of data on the P2P distributed IPFS network, which alleviates the storage pressure of the blockchain and enhances the scalability of the blockchain system. In the IPFS network, large-capacity ciphertext data will be divided into multiple 256 kB packets and stored on different nodes with multiple copies. As long as a node on the IPFS network has a copy of the data content, the data visitors can access it according to the retrieval hash value of the corresponding data. In addition, once the content of the data changes, the address to retrieve the data will change to achieve the antitampering purpose.

5. Experiment and Analysis

5.1. Experimental Environment. To verify the feasibility of the proposed data access control scheme that combines ABE and blockchain technology, a prototype system was constructed for verification. The configuration of the prototype system environment mainly included a blockchain network and IPFS distributed network. This system used the Hyperledger Fabric framework and Docker container technology to build the blockchain network. Due to the limited number of servers, the CentOS 7 virtual machine was used as the running environment of Fabric. This blockchain network consisted of an Orderer node and four Peer nodes. The details are reported in Table 2. The system used go-ipfs to build a private IPFS distributed network, which was composed of seven devices in the same LAN. In addition, a local server was compared with the IPFS distributed network. The specific configuration information of the equipment is presented in Table 3.

5.2. Operation Process of the Smart Contract. In the Hyperledger Fabric framework, a smart contract is called a Chaincode, which is a piece of code written in the Go programming language. It runs in an independent and secure Docker container and initializes and manages the

Table 2: Fabric network configuration.

Node name	Docker image file	Simulation object
orderer.example.com	Hyperledger/Fabric-orderer	Orderer
peer0.org1.example.com	Hyperledger/Fabric-peer	Peer1 node
peer1.org1.example.com	Hyperledger/Fabric-peer	Peer2 node
peer0.org2.example.com	Hyperledger/Fabric-peer	Peer3 node
peer1.org2.example.com	Hyperledger/Fabric-peer	Peer4 node
ca.example.com	Hyperledger/Fabric-ca	CA server
Couchdb	Hyperledger/Fabric-couchdb	Couchdb
Cli	Hyperledger/Fabric-tools	Client

Table 3: Performance parameters of experimental equipment.

Equipment name	Configuration
Local server	Ubuntu Linux release 18.04
Virtual machine	CentOS Linux release 7.8.2003(Core)
IPFS_1	Windows10; CPU i7-8550U; RAM 16.0 GB
IPFS_2	Windows10; CPU r7-4800U; RAM 16.0 GB
IPFS_3	Windows10; CPU i5-9300H; RAM 16.0 GB
IPFS_4	Windows10; CPU i7-8650U; RAM 16.0 GB
IPFS_5	Windows10; CPU i5-6200U; RAM 4.0 GB
IPFS_6	Windows10; CPU i5-4210H; RAM 8.0 GB
IPFS_7	Windows10; CPU i5-9300H; RAM 8.0 GB

ledger state via the transaction submitted by the application. A smart contract works automatically. Once the smart contract is verified, the verified result set is sent to the Orderer nodes, and the changes in the running results will be shared or synchronized to all Peer nodes in the Fabric network. Hyperledger Fabric provides four basic commands to manage the life cycle of smart contracts, namely, package, install, instantiate, and upgrade.

In this experiment, there were two main smart contracts, namely, storage and query smart contracts. The storage smart contract primarily stores the file hash result encrypted by SHA256 (hashfile), the file hash value generated by IPFS (hashipfs), and the access control policy on the blockchain. The query smart contract mainly extracts metadata, such as hashfile and hashipfs. The specific steps are as follows:

(1) Start the Fabric network: use the Docker-Compose tool to read the configuration file to start the Orderer, Peer1, Peer2, Peer3, Peer4, CA, and Couchdb services.
(2) Create a channel: based on the Orderer node, establish a channel named mychannel, and add four Peer nodes to the channel.
(3) Install and initialize the smart contracts: use the install command to install the storage and query smart contracts for Peer nodes, and set the endorsement policy. In our experiment, there are two organizations, and a transaction is valid only when the members of the two organizations jointly endorse.
(4) Use smart contracts: use the fabric-sdk-java tool to call the smart contracts to meet the needs of storage and query.

5.3. Experimental Results

5.3.1. Access Control Structure. The efficiency of the access control scheme is limited by the access control structure and file storage mode. The access control structure in this study consists of the process of attribute-based encryption and file encryption. Therefore, comparative experiments were carried out under different numbers of attributes. All the files in the experiment are numerical data files, which are randomly generated by computer software. The numerical value has no effect on the experimental results. The reported experimental results are the average results of 50 experimental runs.

This experiment was conducted to investigate the time taken by the access control scheme to encrypt and decrypt files under different numbers of attributes. The fixed file size in this experiment was 500 kB. The comparison method was C-AB/IB-ES which was proposed by Hao Wang and Song [24]. C-AB/IB-ES uses attribute-based encryption (ABE) and identity-based encryption (IBE) to encrypt data and uses identity-based signature (IBS) to implement digital signatures. The access control structure of C-AB/IB-ES is more complex than our scheme. As shown in Figure 6, the experimental results reveal that, with the increase of the number of attributes, the encryption and decryption time of the two algorithms increased linearly, which demonstrates that the computational overhead of these schemes were controllable. The encryption and decryption time of our method is much shorter than that of [24]. This indicates that the more complex the access control structure of the CP-ABE algorithm, the more time it takes for access control. Therefore, while pursuing fine-grained data access control, the fewer the number of attributes, the lesser the computational overhead of the system.

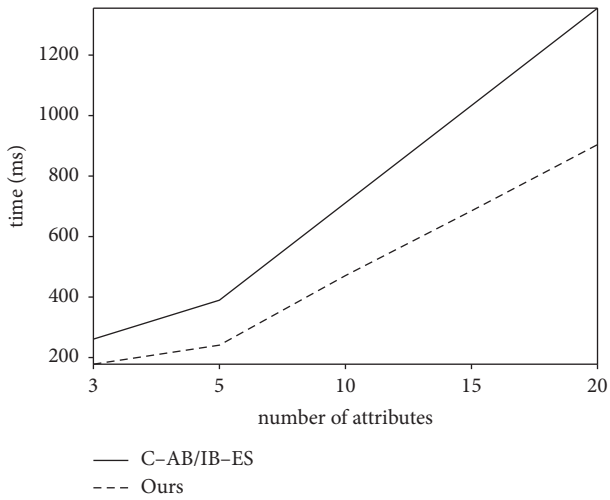

Figure 6: Encryption and decryption times with different numbers of attributes.

5.3.2. File Storage Mode. This experiment was a storage performance comparison between different file storage modes. The access control scheme proposed by Wang et al. [25] uses a cloud server to store files. A local LAN server was used for cloud server storage, and the cluster devices of other IPFS distributed networks were also in the same LAN; the detailed configuration is presented in Table 3. This experiment was conducted to test the time taken to upload and download different-sized files under two storage modes. As shown in Figures 7 and 8, the experimental results indicate that, under the condition of the same file size, the upload and download times of the IPFS storage scheme were, respectively, about 8% and 11% those of the cloud storage scheme. Therefore, the adoption of the proposed IPFS storage scheme can improve the computational efficiency and reduce the system overhead.

5.3.3. Overall Encryption Efficiency. In this experiment, we compare the overall encryption efficiency. The overall encryption time includes file storage time, time spent writing data to the blockchain, CP-ABE encryption time, and AES encryption time. The number of attributes was unchanged, and the encryption and decryption time of different schemes were calculated by changing the size of the files. As shown in Figure 9, with the increase of the file size, the encryption time of Wang's increased rapidly. Both Wang's and our scheme use the CP-ABE and AES algorithm to archive access control, but the two access control processes are different. The overall encryption efficiency of Wang's was lower than that of our scheme.

5.4. Security Analysis. In our scheme, the owner generates and distributes the attribute private key for the user. It not only solves many security risks caused by untrusted attribute authorities in attribute-based encryption schemes but also fine-grained access control is implemented for users without relying on any third party.

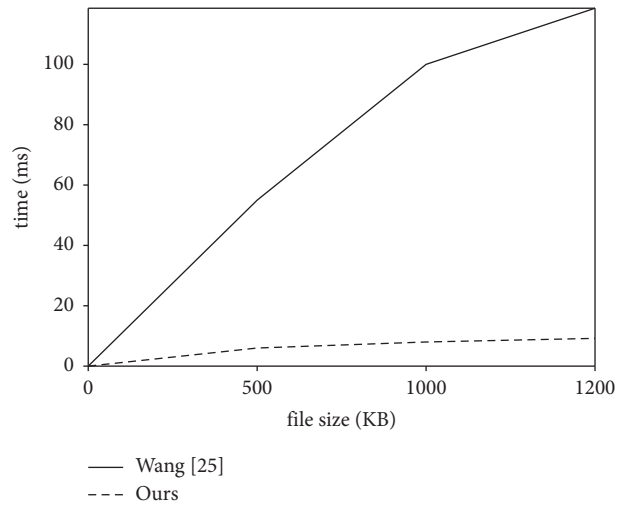

Figure 7: Upload times of the two schemes.

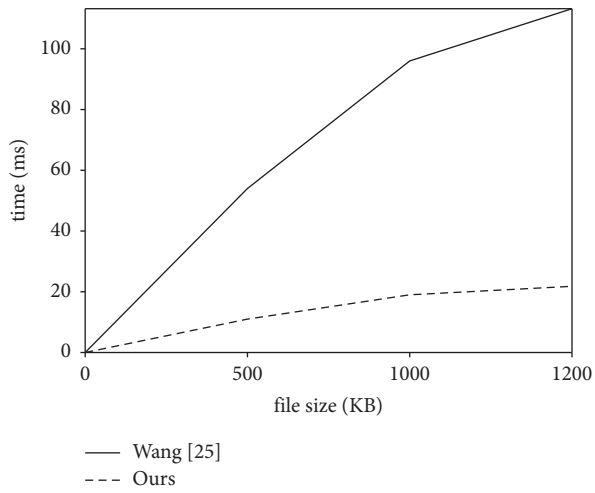

Figure 8: Download times of the two schemes.

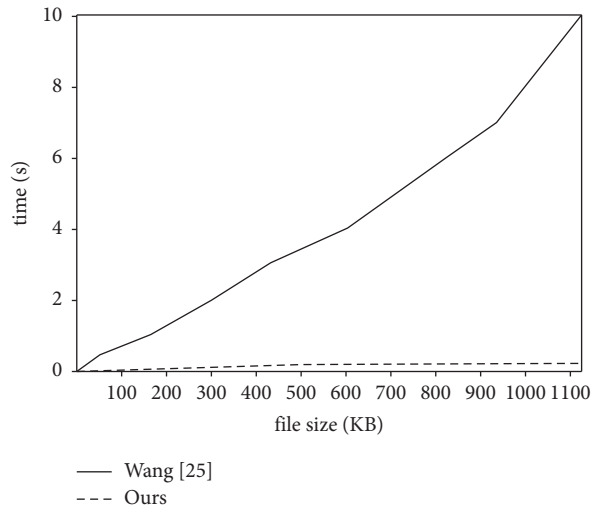

Figure 9: Overall encryption efficiency.

Table 4: Scheme comparison.

Literature	Time	Confidentiality	Integrity	Scalability	Reliability/robustness	Access control
[18]	2017	Strong	Strong	Weak	Weak	Strong
[8]	2018	Strong	Strong	Weak	Weak	Weak
[19]	2018	Strong	Strong	Weak	Weak	Strong
[20]	2018	Strong	Strong	Weak	Weak	Strong
[22]	2018	Strong	Strong	Strong	Weak	Weak
[12]	2019	Strong	Strong	Strong	Weak	Weak
[21]	2019	Strong	Strong	Weak	Weak	Strong
[7]	2020	Strong	Strong	Weak	Weak	Weak
[23]	2020	Strong	Strong	Strong	Strong	Strong
Our	2021	Strong	Strong	Strong	Strong	Strong

Table 4 presents a comparison between the proposed scheme and methods put forward in other similar studies. Ge et al. [7] and Li et al. [8] only applied blockchain technology to their respective schemes, and stored important data on the blockchain to ensure data security and personal privacy. Novo [19], Li et al. [20], Ding et al. [21], and Jemel and Serhrouchni [18] did not only store important data on the blockchain but also demonstrated strong access control ability. Similar to the work reported in the present article, Jemel introduced CP-ABE encryption technology to achieve the fine-grained access control of blockchain data, and the built-in encryption technology ensures the confidentiality and integrity of the data. However, the storage bottleneck of blockchain systems was not considered, and the scalability is weak.

Zhang and Wang [12], Dai et al. [22], and Cheng et al. [23] considered the storage bottleneck of blockchain systems. Zhang and Dai reduced the storage space of blockchain data and improved the scalability of the blockchain system by compressing the data itself. The method proposed by Cheng et al. [23] was also demonstrated to achieve high performance in many aspects. It uses AES to encrypt data and then stores the hash values on the chain to ensure the confidentiality and integrity of the data. Moreover, it stores encrypted data on edge servers with multiple copies, thereby improving the scalability, reliability, and robustness of the system. However, in this scheme, data consumers need the consent of the data owner to access the data, and the fine-grained access ability is weak. With the increase of the amount of access, the communication overhead increases significantly, and the feasibility of practical application is low. Via comparative analysis, the scheme proposed in the present study demonstrated advantages in ensuring data confidentiality, integrity, and scalability.

6. Conclusion

To achieve secure IoT data sharing and fine-grained access control, this paper proposed a data access control scheme based on the CP-ABE algorithm and blockchain technology. In this scheme, the hash value of the data, the location information of the encrypted data, and the access control strategy are stored on the blockchain. The blockchain ensures the integrity and tamperability of these data, so as to achieve fine-grained access to data efficiently. Our scheme not only ensures that the data cannot be tampered with but also ensures that the access control strategy cannot be tampered with.

The proposed scheme alleviates the storage pressure and effectively improves the scalability of the blockchain. The results of a contrast experiment with the cloud storage scheme demonstrate that, as compared to the cloud storage scheme, the proposed scheme consumed 8% of the time to store files and 11% of the time to access files. In addition, a fine-grained access control mechanism that combines the symmetric encryption algorithm and CP-ABE was proposed. In this mechanism, the symmetric encryption algorithm is first used to encrypt the data, and the CP-ABE algorithm is then used to encrypt the symmetric encryption key. This mechanism ensures the security of IoT data, enables data owners to control their data visitors in a fine-grained way, and ensures that the data cannot be tampered with.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by National Key R&D Program of China (Grant no. 2020YFB2104700) and National Natural Science Foundation of China (Grant no. 62136006).

References

[1] Y. Li, Y. Guo, and S. Chen, "A survey on the development and challenges of the Internet of things (IoT) in China," in *Proceedigs of the International Symposium in Sensing and Instrumentation in IoT Era*, September 2018.

[2] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based IoT: challenges," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26–33, 2017.

[3] F. Liu, G. Qian, Y. Yarom, F. Mckeen, and R. B. Lee, "CATalyst: defeating last-level cache side channel attacks in cloud computing," in *Proceedigs of the IEEE International*

Symposium on High Performance Computer Architecture, Barcelona, Spain, March. 2016.

[4] Q. Shao, C. Jin, Z. Zhang, W. Qian, and A. Zhou, "Blockchain technology: architecture and progress," *Chinese Journal of Computers*, vol. 18, no. 8, p. 2449, 2018.

[5] W. Liang, Y. Fan, K. C. Li, D. Zhang, and J. L. Gaudiot, "Secure data storage and recovery in industrial blockchain network environments," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 99, p. 1, 2020.

[6] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[7] C. Ge, Z. Liu, and L. Fang, "A blockchain based decentralized data security mechanism for the Internet of Things," *Journal of Parallel and Distributed Computing*, vol. 141, pp. 1–9, 2020.

[8] H. Li, L. Zhu, M. Shen, F. Gao, X. Tao, and S. Liu, "Blockchain-based data preservation system for medical data," *Journal of Medical Systems*, vol. 42, no. 8, p. 141, 2018.

[9] Y. Fan, S. Liu, X. Lei, K. C. Li, and G. Tan, "One enhanced secure access scheme for outsourced data," *Information Sciences*, pp. 230–242, 2020.

[10] Y. Fan, S. Liu, G. Tan, and F. Qiao, "Fine-grained access control based on trusted execution environment," *Future Generation Computer Systems*, pp. 551–561, 2018.

[11] A. Sahai and B. R. Waters, "Fuzzy identity-based encryption," in *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*, Berlin, Heidelberg, May 2005.

[12] G. Zhang and R. Wang, "Blockchain shard storage model based on threshold secret sharing," *Journal of Computer Applications*, vol. 39, no. 9, pp. 2617–2622, 2019.

[13] Blockchain, *The Blockchain Data of Bitcoin [EB/OL]*, Blockchain, Luxembourg, UK, 2018, https://www.blockchain.com/.

[14] B. Confais, A. Lebre, and B. Parrein, "An object store service for a fog/edge computing infrastructure based on IPFS and scale-out NAS," in *Proceedigs of the IEEE International Conference on Fog & Edge Computing*, 2017.

[15] Q. Xu, Z. Song, R. Goh, and Y. Li, "Building an Ethereum and IPFS-based decentralized social network system," in *Proceedigs of the 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, Singapore, December 2018.

[16] M. Klems, J. Eberhardt, S. Tai, S. Härtlein, S. Buchholz, and A. Tidjani, "Trustless intermediation in blockchain-based decentralized service marketplaces," in *Proceedigs of the International Conference on Service-Oriented Computing*, Malaga, Spain, November 2017.

[17] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedigs of the IEEE Symposium on Security & Privacy*, 2007.

[18] M. Jemel and A. Serhrouchni, "Decentralized access control mechanism with temporal dimension based on blockchain," in *Proceedigs of the IEEE International Conference on E-business Engineering*, November 2017.

[19] O. Novo, "Blockchain meets IoT: an architecture for scalable access management in IoT," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.

[20] D. Li, P. Wei, W. Deng, and F. Gai, "A blockchain-based authentication and security mechanism for IoT," in *Proceedigs of the 2018 27th International Conference on Computer Communication and Networks (ICCCN)*, Hangzhou, China, October 2018.

[21] S. Ding, J. Cao, C. Li, K. Fan, and H. Li, "A novel attribute-based access control scheme using blockchain for IoT," *IEEE Access*, vol. 7, p. 1, 2019.

[22] M. Dai, S. Zhang, H. Wang, and S. Jin, "A low storage room requirement framework for distributed ledger in blockchain," *IEEE Access*, vol. 6, pp. 22970–22975, 2018.

[23] G. Cheng, Z. Huang, and S. Deng, "Data management of Internet of things based on blockchain and edge computing," *Chinese Journal on Internet of Things*, vol. 4, no. 2, pp. 2–10, 2020.

[24] H. Wang and Y. Song, "Secure cloud-based EHR system using attribute-based cryptosystem and blockchain," *Journal of Medical Systems*, vol. 42, no. 8, p. 152, 2018.

[25] S. Wang, D. Zhang, and Y. Zhang, "Blockchain-based personal health records sharing scheme with data integrity verifiable," *IEEE Access*, vol. 7, pp. 102887–102901, 2019.

[26] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS 2006, Alexandria, VA, USA, November 2006.