# StorSec: A Comprehensive Design for Securing the Distributed IoT Storage Systems

Shiwen Zhang, Wen Zhang, Wei Liang, *Senior Member, IEEE*, Wenqiang Jin, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Internet of Things (IoT) networks have penetrated our daily life and industries. However, IoT devices are typically small-sized with constrained storage. Distributed storage systems are emerging as promising solutions to tackle such challenges. InterPlanetary File System (IPFS) is a desired framework enabling IoT devices to upload its data to a distributed cloud while returning a hash-ID for downloading and file-sharing purposes. Nevertheless, IPFS lacks of robust security design and is vulnerable to security threats such as data tampering, and data leakage. In particular, whenever device *A*'s file hash-ID is shared to an arbitrary device *B*, device *A* will fully lose the control over file. In other words, device *B* could further share it to anyone without device *A*'s agreements. To conquer the challenge, we propose a comprehensive design for securing the distributed IoT storage systems, named StorSec. Specifically, we design a new heterogeneous framework using an improved attribute encryption algorithm to eliminate the single-point performance bottleneck problem, which not only realizes fine-grained access control and ensures the security of data during transmission, but also improves the performance of key generation. Secondly, we design an anomaly detection algorithm, which is based on hashchain technology and combines the user privacy metadata stored on the blockchain to complete the verification process, effectively protecting the file hash identifier, ensuring access control to the file, and thus providing protection for the security and integrity of data storage. Furthermore, we design an auditing algorithm that helps the system in tracking malicious entities. Ultimately, the security and efficiency of the proposed scheme are evaluated by both security analysis and experimental results.**

***Index Terms*—IoT, blockchain, IPFS, attribute-based encryption, verifiable.**

## I. Introduction

THE INTERNET of Things (IoT) has grown to be an essential component of our everyday existence. However, the storage capacity of IoT devices is facing serious limitations. According to predictions from the International Data Corporation (IDC) [1], nearly 80ZB of data will be generated by IoT devices by 2025, while many IoT devices may only have a few MB to GB of storage space. This means that these devices cannot store all the data generated locally, which undoubtedly brings some challenges. The vast volumes of data produced by IoT requires the use of hardware virtualization technology for storage, and this further stimulates the development of centralized storage and distributed storage technologies. The typical representative of centralized storage technology is cloud storage. Many organizations or commercial companies typically use the cloud [2], [3], [4] to store data. However, the user data generated by the terminal devices often involve personal privacy information. Delivering this information to third-party cloud data centers not only reduces the efficiency of data processing due to high transmission latency, but also increases the risk of privacy leakage [5], [6], [7], as shown in Fig. 1.

To solve the aforementioned problems in centralized storage, the concept of distributed storage [8], [9], [10] for the IoT environment has gradually emerged. IPFS [11], with its peer-to-peer global distributed file system, offers a robust solution for handling massive data storage. It eliminates single points of failure through distributed and persistent storage, supports deduplication, high throughput, and content addressing. However, although IPFS offers high storage efficiency, it does not provide security for data by itself and is vulnerable to security threats such as data tampering and data leakage. This means that without additional security measures, the stored content could be accessed by unauthorized users.

Due to the risks of malicious tampering and privacy leakage during data storage and transmission, many researchers have recommended encrypting data and enforcing access control before transmission to prevent unnecessary network attacks [12], such as replay attacks, collusion attacks, man-in-the-middle attacks, data tampering, and DDoS attacks. Many ideas have been put out to address the encryption challenge of data access control [13], [14], [15]. Of them, one of the most potential is Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [15]. It is capable of guaranteeing both confidentiality and detailed access control. At present, access control schemes based on CP-ABE are primarily divided into two categories: single-authority [16], [17], [18] and multi-authority [19], [20], [21]. In the existing single-authorization scheme, only one authoritative organization is responsible for

Received 22 September 2024; revised 17 May 2025; accepted 7 August 2025. Date of publication 11 August 2025; date of current version 5 December 2025. This work was supported in part by the Scientific Research Fund of Hunan Provincial Education Department under Grant 24A0337, and in part by the Natural Science Foundation of Hunan province under Grant 2025JJ50348. The associate editor coordinating the review of this article and approving it for publication was P. Varga. *(Corresponding author: Wenqiang Jin.)*

Shiwen Zhang, Wen Zhang, and Wei Liang are with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China (e-mail: shiwenzhang@hnu.edu.cn; francefff@qq.com; wliang@hnust.edu.cn).

Wenqiang Jin is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: wqjin@hnu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TNSM.2025.3597550

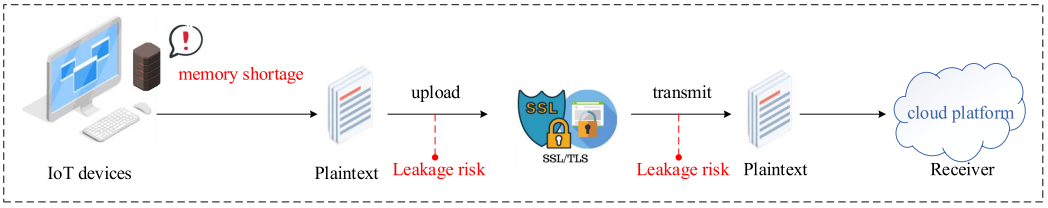


Fig. 1. The problem of IoT data storage.

allocating all attributes. In situations where user volume is high, a single point of performance bottleneck can easily occur, causing system congestion. Since all attributes are managed by one authoritative organization, users may need to queue when obtaining private keys, resulting in low key generation efficiency. In multi-authorization application scenarios, if multiple authoritative institutions manage different subsets of attributes separately, this may also lead to inefficient key distribution. Moreover, this situation also presents the issue of data breaches resulting from coordinated attacks by entities. The existing schemes [22], [23], [24] propose some accountability mechanisms for collusion among entities but do not offer the corresponding technologies to detect collusion activities of entities to prevent data from being illegally obtained.

In this article, we propose a comprehensive design scheme to ensure the security of distributed IoT storage systems. First, we build a decentralized storage framework based on the blockchain, using IPFS to establish verifiable links between blockchain status and content addresses published to IPFS, ensuring the integrity and correctness of data. Simultaneously, the improved adaptability and detailed features of the CP-ABE scheme are used for data encryption. This technology not only improves the performance of key generation, but also ensures the security of data in transit while realizing fine-grained access control. Then, to detect abnormal behaviors of institutions or users and prevent data from being illegally obtained, this scheme designs a private key verification algorithm based on the hashchain technique, which combines with the metadata of user's private data stored on the blockchain to verify the authenticity of the user's private key. Next, we design an audit algorithm that can help the system track malicious entities. Finally, we conduct extensive simulation experiments to confirm the security and efficiency of our proposed design. The primary contributions of this article are summarized as below.

(1) We design a new heterogeneous framework using an improved attribute encryption algorithm to eliminate the single-point performance bottleneck problem, which not only realizes fine-grained access control and ensures the security of data during transmission, but also improves the performance of key generation.
(2) We propose a verifiable multi-authorization attribute encryption scheme based on blockchain, and use blockchain to design an anomaly detection algorithm that prevents data from being illegally accessed by regulating the anomalous behaviors of institutions or users.
(3) We propose a blockchain-based decentralized auditing algorithm that helps the system track down malicious entities while being able to ensure the transparency of the auditing process and the immutability of the data.
(4) We conduct a detailed security analysis, proving that the scheme is secure under the Decisional q-parallel Bilinear Diffie-Hellman Exponent Assumption(q-BDHE) model, and can resist collusion attacks from AAs and users. The sufficient experimental results demonstrates that this scheme exhibits greater efficiency.

The structure of the remaining sections of this article is outlined as follows. In Section II, we reviewed the related work of this article. Section III introduces the relevant preliminaries. The system model and workflow are in Section IV. We describe the proposed scheme in Sections V and VI. In Section VII, we present a safety analysis of this scheme. In Section VIII, we assess the performance of the scheme through the experimental results. The article concludes with a summary of our findings in Section IX.

## II. Related Work

### A. Secure Data Storage for IoT

In recent years, secure data storage has always attracted wide attention from the IoT industry and academia. For example, Chen et al. [25] proposed decentralized public auditing scheme based on blockchain technology to address the security challenges of data integrity in cloud storage. This scheme aims to decentralize and entrust the auditing tasks to multiple Cloud Service Providers (CSPs), and uses blockchain technology to record the entire auditing process to ensure the security and integrity of data. However, the CSPs mentioned in the article are set as semi-trusted, and the uncertainty of their behaviors poses potential risks to data storage. Specifically, once the outsourced data is damaged during the management or processing by a particular CSP, it may cause a single point of failure, thereby leading to data loss or integrity damage. Lu et al. [26] designed a novel anonymous authentication scheme with a group signature priority that minimizes both computational and communication overheads. They built a blockchain-based IoT sensor cloud storage protocol using this proposed group signature scheme, and implemented proxy re-encryption to ensure the security of data transmission. Although this scheme achieved secure data storage and sharing in cloud-supported IoT environments, it still faces some single-point failure issues, including the possibility of cloud service disruptions and internal malicious attacks. These potential risks could potentially undermine the stability and security

of the system. Later, Azbeg et al. [27] proposed a decentralized IoT healthcare system solution based on blockchain technology, which addresses the security issues faced by traditional cloud storage or third-party storage. The scheme ensures the security and tamper-proof nature of data by utilizing blockchain technology, while also employing IPFS for distributed storage. However, IPFS itself does not provide data security, and it is susceptible to threats such as data tampering and data leakage. The integration of blockchain technology with IPFS ensures the integrity of the data. Nevertheless, the inherent openness of blockchain allows anyone to view the content identifiers (hash-ID) returned by IPFS. This means that without additional security measures, unauthorized users may gain access to the stored content. To further protect data, Ullah et al. [28] introduced a blockchain-based solution for decentralized distributed storage and sharing. This solution integrates end-to-end data encryption, the IPFS decentralized storage system, and the Ethereum blockchain. By encrypting the content identifiers stored in IPFS, the data storage is effectively safeguarded. However, in large-scale applications, the use of symmetric encryption for file data poses challenges in key management and distribution. Symmetric encryption requires each pair of users to have an independent key to ensure the security of communication. In a network composed of $N$ users, the number of keys that need to be managed grows exponentially. This complexity in key management not only increases the operational costs of the system but also may become a potential security risk point, as issues such as key loss, leakage, or improper management can all lead to the failure of data protection. To address these challenges, we propose a decentralized storage solution that simultaneously ensures data confidentiality, minimizes key management overhead, and prevents metadata leakage, making it well-suited for secure and scalable storage in IoT applications.

### B. Multi-Authority Attribute-Based Encryption

In order to prevent data leakage during storage and transmission, we summarized the following research on data encryption and access control simultaneously. Currently, attribute-based encryption technology CP-ABE is seen as one of the most potential data access control technologies [15]. According to the number of participating institutions, these CP-ABE schemes can be classified into two categories: single authority-based schemes and multiple authority-based schemes. In a single-authority application scenario, the distribution of keys corresponding to attributes is provided by a one institution center. The computation is not only heavy, but we also have to have faith in the one authorization hub. To address the issue of low efficiency and poor security of the single authorization center, multiple authorization alternatives have emerged. In 2007, Chase in [19] proposed the concept of attribute-based encryption scheme based on multiple authorities. However, in multi-authority application scenarios, there is no corresponding technology to detect the abnormal behavior of the responsible party, to prevent data from being illegally obtained. Yang et al. in [29] proposed a secure data access control scheme enabled by edge blockchain with a fair accountability mechanism for data sharing in smart grid. However, this scheme did not explain in detail how to determine whether the data was leaked and how to identify the responsible party for the leakage. Bader et al. in [30] presented a blockchain-based scheme that preserves privacy for supporting lightweight multi-hop information accountability in supply chains. The scheme designed an accountability mechanism based on smart contracts, which can be used to punish and incentivize negligent and irresponsible behaviors in supply chains. However, this literature [30] did not provide detailed explanations of the design and implementation details of the smart contracts, nor did it verify their correctness and robustness, so it is difficult to ensure that they can effectively execute the accountability mechanism and avoid logical errors or malicious attacks.

In contrast to prior research, this article proposes a design for a blockchain-based distributed data storage system. This system designs a new heterogeneous framework with improved attribute encryption algorithms to eliminate the single-point performance bottleneck problem, which not only realizes fine-grained access control and ensures the security of data during transmission, but also improves the performance of key generation. To prevent unauthorized access to data, we introduce an anomaly detection algorithm. This algorithm leverages hashchain technology combined with user privacy data metadata stored on the blockchain to complete the verification process, effectively protecting file hash identifiers and ensuring control over the files, thereby guaranteeing the security and integrity of data storage. Moreover, we design an audit mechanism that can help the system track the responsible parties.

## III. Preliminaries

### A. InterPlanetary File System (IPFS) for IoT

The swift advancement of the IoT has resulted in a marked escalation in the necessity for data storage capacity. This growth has driven the development of storage solutions such as centralized and distributed storage. Centralized storage is easy to manage and maintain, but it also faces challenges in terms of data security and efficiency. Distributed storage technology, with its better security and scalability, is considered the future direction of IoT development. IPFS [11] is a distributed file system that operates on a peer-to-peer basis, enabling content-addressable storage. Multiple nodes running the IPFS program constitute the IPFS storage network, and the data stored in this network is divided into multiple blocks and stored at different nodes. When a user uploads a file to an IPFS node for storage, the node will store the file after dividing it into blocks. Each file block is organized in the form of a Merkle Directed Acyclic Graph (Merkle DAG), as shown in Fig. 2, and the root hash of the Merkle DAG is used to represent the file. At the same time, a Distributed Hash Table (DHT), as shown in Fig. 3 is used to implement the location of file content through hash values. IPFS supports deduplication and content addressing, but due to the lack of effective data integrity and security measures, there may be issues with data reliability and security.

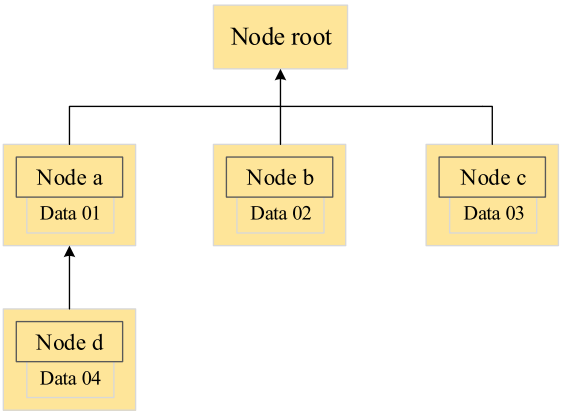


Fig. 2. Merkle DAG.

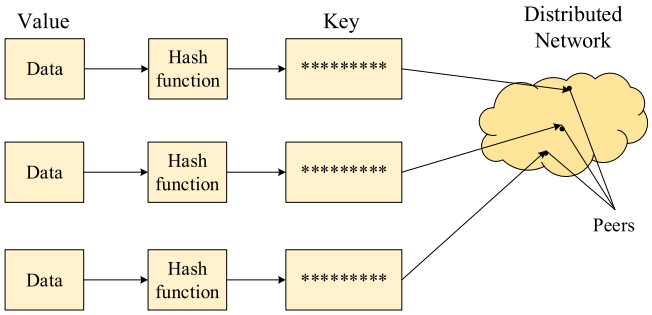


Fig. 3. DHT.

### B. Research Problem

The problem we consider is how to design a secure data storage solution in the background of the IoT. We focus on how to ensure its security during data transmission, that is, using reliable encryption technologies to prevent data from being intercepted or tampered with during transmission to prevent unauthorized access. Simultaneously, we take into account the following potential threats.

Threat 1: In multi-authority encryption scenarios, Attribute Authorities (AAs) are sincere and curious. They will abide by the contract, assign attributes to users, and verify their legitimacy. Since the verification of users is done manually, there may be careless mistakes or malicious behavior, leading to illegal access to data by users.

Threat 2: AAs and users may be malicious, and they might initiate a collusion attack, attempting to conspire to obtain and decrypt ciphertexts that neither can decrypt alone.

## IV. StorSec Design

In this section, we describe the system model and workflow of the proposed scheme. Table I lists the key symbols used in this article.

### A. System Model

This article proposes a blockchain-aided verifiable multi-authority attribute-based encryption scheme for distributed IoT Storage System. It consists of six entities, namely, certificate authority (CA), attribute authorities (AAs), data owner (DO), data user (DU), interplanetary file system (IPFS), and blockchain (BC), as shown in Fig. 4. To address security issues during data transmission, we set up a multi-authority attribute encryption framework consisting of a single *CA* and multiple *AA*s. We define each of these entities as follows.

TABLE I
Key Notations

| Notation | Description |
|---|---|
| $CA$ | certificate authority |
| $AAs$ | attribute authorities |
| $\lambda, p, G_g$ | security parameter, prime order, multiplicative cyclic group |
| $PK, msk$ | system public and private keys |
| $SK$ | private key |
| $hashchain$ | key chain |
| $H$ | secure one-way hash function |
| $E_k$ | symmetric encryption function |
| $x$ | randomly selected values |
| $M$ | Plaintext data |
| $CT$ | Ciphertext |
| $\triangle T$ | allowable time range |
| $CID$ | a hash value generated from the content, noted as a content identifier |
| $GID$ | global identifier |

1) CA: The *CA* serves as the manager of the entire framework and is responsible for system initialization. It is considered a trusted entity within the system, forming the foundation for secure and authenticated operations.

2) AAs: *AAs* are responsible for verifying the legitimacy of user identities. In a multi-authority attribute-based encryption setting, each *AA* governs a subset of attributes and issues attribute-based keys to users based on their verified credentials.

3) DO: The *DO* defines the access policy for each file and encrypts the data accordingly before outsourcing it.

4) DU: A *DU* is any entity requesting access to the data. Only users who have been verified by the blockchain are authorized to retrieve the *CID* from *BC* and then fetch the corresponding ciphertext from IPFS.

5) IPFS: The encrypted data is stored within IPFS, and we utilize IPFS to establish a verifiable link between the *BC* state and the content address published to IPFS, ensuring the integrity of the data.

6) Blockchain: The blockchain component stores metadata related to users' private data and manages access control information. To mitigate data leakage risks from collusion attacks in multi-authority ABE, the blockchain uses hashchains to provide tamper-evident verification of stored metadata.

### B. Workflow

The main workflow is presented in Fig. 5. The scheme consists of two parts: **Multi-ABE** and **BC Verify**.

In the **Multi-ABE** process, we describe it in four stages.

Stage 1: during the system initialization process, the *CA* is responsible for generating the public key and a series of parameters, and assigns unique identifiers when the *DU* and *AA*s register.

Stage 2: when encrypting data, *DO* first encrypts the data with a symmetric encryption algorithm. Then, it secures the resulting key by encrypting it with the public key acquired

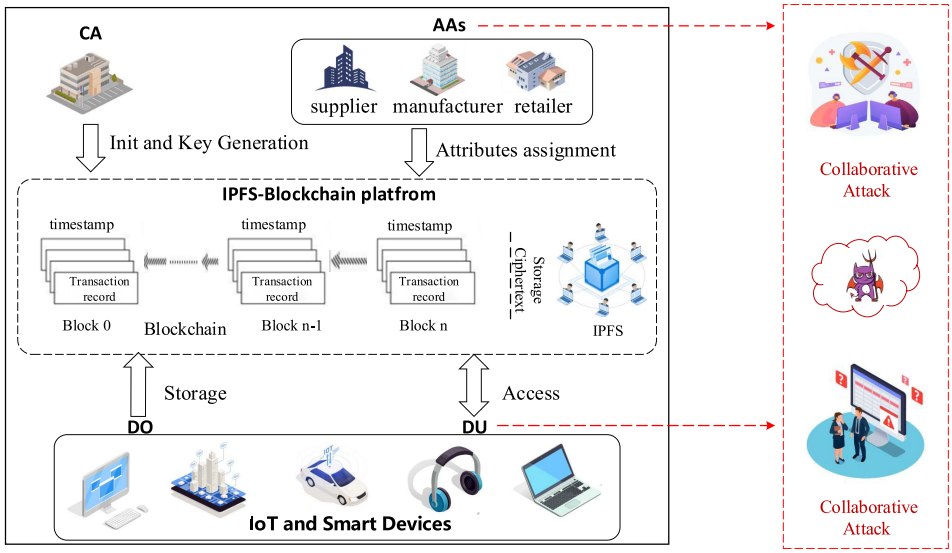


Fig. 4. System model.

from *CA*. The owner then dispatches both the encrypted data and the symmetric key (in the form of ciphertext *CT*) to IPFS for storage.

Stage 3: in the key management process, the *CA* generates the corresponding keys for the users and uploads the metadata to the *BC*.

Stage 4: when users are authenticated through the correct key and their attribute set matches the access structure in the ciphertext, they can obtain the ciphertext and decrypt it using their private key.

In the **BC Verify** process, users can compare the hash value of the data before decryption to verify whether the data has been tampered with. To prevent data leakage caused by collusion of malicious entities, we utilize blockchain technology for anomaly detection. Specifically, before obtaining the ciphertext, the *DU* first initiates a keyword request to the *BC*. Upon receiving this request, *BC* finds the corresponding index value in the indextable based on related metadata and sends it to the *CA* to request the return of the related hash value. *BC* performs hash verification with the private key uploaded by *DU*, and after verification, returns the keyword *CID* to *DU*. *DU* uses this keyword to query and download the ciphertext from IPFS.

## V. Multi-ABE

### A. System Initialization

Firstly, let $\mathbb{G}_g$ and $\mathbb{G}_T$ as two multiplicative cyclic groups of prime order $p$, with $g$ serving as the generator of $\mathbb{G}_g$. The bilinear map $e$ is, $e : \mathbb{G}_g \times \mathbb{G}_g \to \mathbb{G}_T$. *CA* selects a safety collision hash function $H{:}\{0,1\}^* \to \mathbb{Z}_p^*$ and a pseudorandom function $PRF : \mathbb{Z}_p^* \times \{1, 2, \ldots, n\} \to \mathbb{Z}_p^*$. The *CA* randomly generates four numbers $\alpha, \beta, u, v \in \mathbb{Z}_p^*$, and for each property $i \in \{1, 2, \ldots, U\}$, where $U$ denotes the maximum number of attributes, it randomly selects the parameters $h_1, \ldots, h_U \in \mathbb{G}_g$. The system master key $msk = \{\alpha, \beta, u, v\}$ is generated, and the public key that needs to be publicized is

$$PK = \left(\mathbb{G}_g, \mathbb{G}_T, H, g, e(g,g)^{\alpha}, g^{\beta}, h_1, \ldots, h_U\right) \quad (1)$$

Then, *CA* begins by processing the registration of *AAs* and users. For each *AA* with identity *Aid*, the *CA* generates a signing and verification key pair: a private key $sk_{Aid} \in \mathbb{Z}p$ and the corresponding public key $pk_{Aid} = g^{sk_{Aid}}$. The *CA* then issues a certificate $cert_{Aid}$ containing $pk_{Aid}$, and securely sends both $cert_{Aid}$ and $sk_{Aid}$ to the respective *AA*. Similarly, each user is assigned a unique identifier *Uid*, and is issued a private key $sk_{Uid}$ along with a certificate $cert_{Uid}$ by the *CA*.

Next, *CA* constructs and initializes a hashchain. Given a hash function *H*, *hashchain* denotes a hashchain, which is a finite sequence of $x_0, x_1, \ldots, x_{l-1}$, satisfying $x_i = H(x_{i-1})$ for $i = 1, 2, \ldots, l$, where $l$ is the length of the hashchain, $x_0$ is the seed of the hashchain, $x_i$ is the *i*-th element in the hashchain, and $x_{l-1}$ is the end of the hashchain. Then we have:

$$hashchain = (x_0 \leftarrow x_1 \leftarrow, \ldots, \leftarrow x_{l-1})$$

The process of system initialization can be summed up as follows using Algorithm 1.

### B. Data Encryption

The data encryption process is performed by the data owner themselves. *DO* first selects a symmetric encryption function $E_k : \{0,1\}^* \to \{0,1\}^*$, such as AES, and CP-ABE encryption algorithm. The *DO* then selects a random symmetric key *k* and uses this key along with the symmetric encryption function $E_k$ to encrypt the plaintext data *M*, resulting in the encrypted data $E_k(M)$. To secure the symmetric key *k*, the *DO* utilize the CP-ABE in accordance with their personally defined access policy $\Gamma$. Consider a set of parties, denoted as $\{P^1, P^2, \ldots, P^n\}$. We define a collection $\Gamma \subseteq 2^{\{P^1, P^2, \ldots, P^n\}}$, as monotone if for all subsets $E$ and $F$, if $E$ is an element of $\Gamma$ and

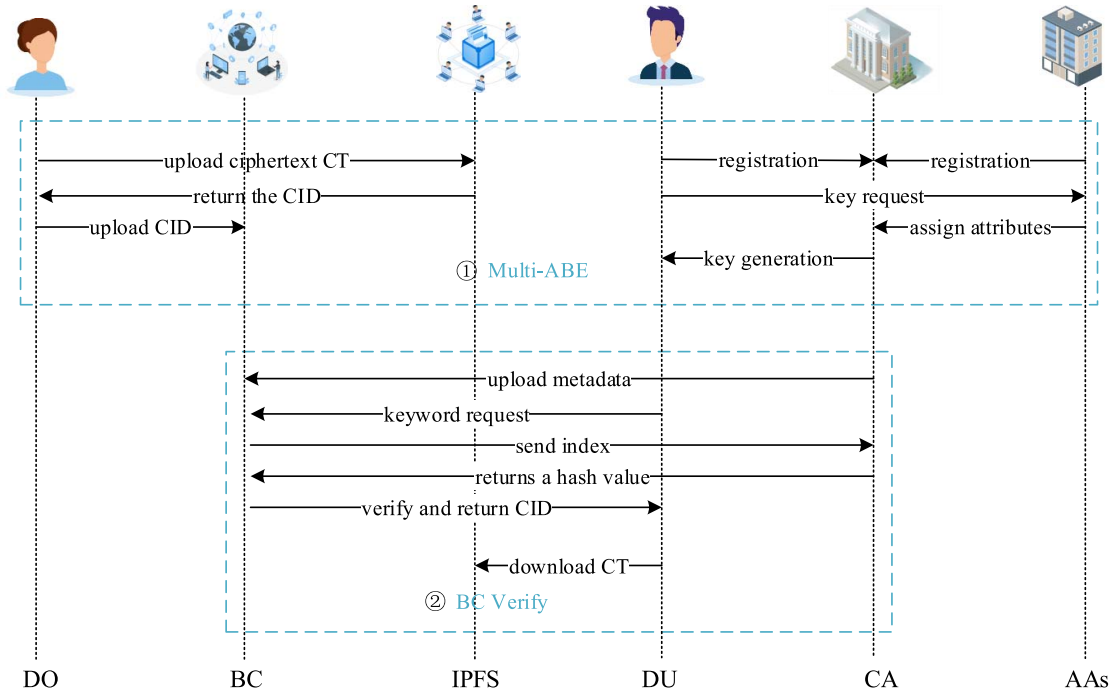


Fig. 5. Workflow.

**Algorithm 1** System Initialization

**Input:** Security parameter $\lambda$, attribute universe $U$;
**Output:** System public key *PK*, master key *msk*;
1: Choose a prime $p$ of bit-length $\lambda$;
2: Let $\mathbb{G}_g$, $\mathbb{G}_T$ be cyclic groups of order $p$, with generator $g \in \mathbb{G}_g$;
3: Define bilinear map $e$:$\mathbb{G}_g \times \mathbb{G}_g \to \mathbb{G}_T$;
4: Define collision-resistant hash function $H$:$\{0,1\}^* \to \mathbb{Z}_p^*$ and pseudorandom function $PRF$:$\mathbb{Z}_p^* \times \{1,...,n\} \to \mathbb{Z}_p^*$;
5: *CA* randomly selects $\alpha, \beta, u, v \in \mathbb{Z}_p^*$;
6: For $i = 1$ to $|U|$, choose $h_i \in \mathbb{G}_g$;
7: Compute $msk = \{\alpha, \beta, u, v\}$;
8: Compute $PK = (\mathbb{G}_g, \mathbb{G}_T, H, g, e(g,g)^\alpha, g^\beta, h_1, ..., h_U)$;
9: **return** *PK*, *msk*;
10: *CA* assigns *Aid* and *Uid*, and sends certificate $cert_{Aid}$ and $cert_{Uid}$;
11: Hashchain construction:
12: Generate random $x_0 \in \mathbb{Z}_p^*$;
13: For $i = 1$ to $l$: $x_i = H(x_{i-1})$;
14: Set $hashchain = (x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_{l-1})$;

$E$ is a subset of $F$, then $F$ is also an element of $\Gamma$. An access structure is defined as a collection of non-empty subsets of$\{P^1, P^2, \ldots, P^n\}$, denoted as $\Gamma$, which is a subset of the power set of $2^{\{p_1,p_2,...,p_n\}}$ excluding the empty set. The subsets that are elements of $\Gamma$ are referred to as authorized sets, while the others are referred to as unauthorized sets. The *DO* formulates a monotonous Boolean expression, which can be transformed into an LSSS access structure represented by $(\mathcal{A}, \varrho)$ using the method introduced in [31]. A matrix $\mathcal{A}$ of dimension $r \times n$ exists. Each row of $\mathcal{A}$ is assigned to a specific attribute using the function $\varrho$, denoted as $\varrho(i) \in \{at_1, at_2, \ldots, at_U\}$. The symmetric key $k$ is encrypted by choosing a random key parameter $s$. *DO* first selects a random vector $\vec{\mu} = (s, a_2, \ldots, a_n) \in \mathbb{Z}_p^n$ to share the secret value $s$. Then, for $i = 1, 2, \ldots, r$, the *DO* calculates $\eta_i = \mathcal{A}_i \vec{\mu}$, where $\mathcal{A}_i$ represent the vector corresponding to the $i$-th row of matrix $\mathcal{A}$. Finally, the *DO* randomly chooses $p_1, p_2, \ldots, p_r \in \mathbb{Z}_p^*$, and the ciphertext is uploaded as $CT_{(M)}$:

$$CT_{(M)} = \Big(C' = ke(g,g)^{\alpha s}, C_0 = g^s \quad \forall i = 1 \text{ to } r,\ C_i^1 = g^{\beta\eta_i} h_{\varrho(i)}^{-p_i}\ C_i^2 = g^{p_i}\Big). \quad (2)$$

Then, *DO* sends the ciphertext to IPFS, IPFS returns a hash value *CID*, and upon receiving it, *DO* uploads the *CID* as a keyword to the blockchain. We can present the process of data encryption storage as Algorithm 2.

### C. Key Management

In our proposed scheme, which differs from traditional key management methods, a user selects an *AA* for verification and attribute allocation. After completing the above steps, the user obtains the key from *CA*, which maps the hash value of the user's private key to the corresponding position in the hashchain. Then, *CA* uploads and maps the position index, the user's attribute set $S_{Uid}$, and *Uid* to the index table on the blockchain.

When sending a key request, the user *Uid* needs to choose an *AA* with the identity *Aid* through a certain scheduling algorithm, and sends $Cert_{Uid}$ to demonstrate the legality of his/her identity. After the verification is successful, *AA* gets the current timestamp value $T_C^A$, computes $t = H(Uid||T_C^A||0)$ and $t' = H(Uid||T_C^A||1)$, and generates

$$D_i = h_i^{pk_{Aid}t},\ d_i = h_i^{t'},\ \forall i \in S_{Uid} \quad (3)$$

**Algorithm 2** Data Encryption and Storage

**Input**: Symmetric key *k*, public key *PK*, data *M* and access policy $\Gamma$;
**Output**: Ciphertext $CT, CID$;
1: *DO* selects a arbitrary key $k \in \mathbb{G}_T$ as the symmetric key, and uses the key to encrypt *M*;
2: Compute $E_k(M)$ using AES or other symmetric encryption;
3: Convert $\Gamma$ into LSSS matrix $(\mathcal{A}, \varrho)$;
4: Run CP-ABE encryption on $k$:
Compute $C' = ke(g,g)^{\alpha s}$, $C_0 = g^s$;
For each row $i$ in $\mathcal{A}$, compute $C_i^1, C_i^2$;
5: Form $CT = (E_k(M), C', C_0, \{C_i^1, C_i^2\})$;
6: Upload *CT* to IPFS;
7: Receive the content identifier *CID* from IPFS;
8: Upload *CID* to the blockchain as the keyword;
9: **return** *CT*, *CID*;

then delivers the subsequent communication to *CA*:

$$\{Uid, Aid, S_{Uid}, D_i, d_i, T_C^A\}$$

Upon receipt of the message from *AA*, *CA* finds the matching $pk_{Aid}$ according to *Aid*. Next, *CA* calculates the difference between the current time and the received time $T_C^A$, determines whether the transmission delay is within the allowable range $\Delta T$, a system-defined security parameter (e.g., 5 seconds). If the delay is acceptable, it proceeds to the next step; otherwise, it will terminate the current operation. *CA* continues to compute $t$ and $t'$, and ensures that they are not reused by the same user. This mechanism helps to prevent collusion attacks from *AAs*. In the end, for the user *Uid*, *CA* generates the key $SK_{Uid}$ using master key *msk* in the manner described below:

$$\begin{aligned} D &= g^{\alpha}(pk_{Aid})^{\beta ut} g^{\alpha\beta t'} = g^{\alpha}\left(g^{sk_{Aid}}\right)^{\beta ut} g^{\alpha\beta t'}, \\ D_0 &= (pk_{Aid})^{ut} g^{\alpha t'} = \left(g^{sk_{Aid}}\right)^{ut} g^{\alpha t'}, \\ &\forall i \in S_{Uid}, D_i' = D_i^u \cdot g^{v(t+t')} = h_i^{sk_{Aid}\, ut} \cdot g^{v(t+t')}, \\ D_i'' &= d_i^{\alpha} \cdot g^{-v(t+t')} = h_i^{\alpha t'} \cdot g^{-v(t+t')}. \end{aligned} \tag{4}$$

To simplify the formula, we have introduced a parameter $w$, where $w = sk_{Aid}\, ut + \alpha t'$. Consequently, the user's secret key formulations, $SK_{Uid}$, can be expressed as follows:

$$\begin{aligned} &D = g^{\alpha} g^{\beta w},\ D_0 = g^{w}, \\ &\forall i \in S_{Uid}, D_i' = D_i^u \cdot g^{v(t+t')} = h_i^{sk_{Aid}\, ut} \cdot g^{v(t+t')}, \\ &\qquad D_i'' = d_i^{\alpha} \cdot g^{-v(t+t')} = h_i^{\alpha t'} \cdot g^{-v(t+t')}. \end{aligned} \tag{5}$$

*CA* securely distribute user attribute private keys.

$H$ represents a hash function, which maps data items to the interval $[0, l-1]$, that is, $H : d \rightarrow [0, l-1]$, where $d$ represents the set of data items, i.e., $d = \{H(SK_{Uid_1}), H(SK_{Uid_2}), \ldots, H(SK_{Uid_l})\}$. The CA computes $index = H(SK_{Uid}) \mod l$ and securely maps the user's private key hash $H(SK_{Uid})$ to the corresponding index in the hashchain: $hashchain[index] = H(SK_{Uid})$. An index table is maintained on the blockchain, which maps each tuple $(Uid, S_{Uid})$ as keys to the value *index*. This index indicates the position of the user's private key hash in the hashchain and ensures verifiability. The data format stored in the index table is shown in Fig. 6. The process of key management uses Algorithm 3, which can be described as follows.

| $S_{Uid_1}$ & $Uid_1$ | $S_{Uid_2}$ & $Uid_2$ | $S_{Uid_i}$ & $Uid_i$ |
|---|---|---|
| 0 | 1 | ... |

Fig. 6. The data format stored in the index table.

**Algorithm 3** Key Management

**Input**: Master key *msk*, user attribute set *S*, and allowable time difference $\Delta T$;
**Output**: User private key *SK*;
1: CA selects an *AA* for user *Uid*;
2: *AA* receives request and computes $t$ and $t'$;
3: Compute $D_i = h_i^{pk_{Aid}\, t}$ and $d_i = h_i^{t'}$;
4: Send $(Uid, Aid, S_{Uid}, D_i, d_i, T_C^A)$ to *CA*;
5: *CA* obtains the current time $T'$;
6: Compute $\triangle T' = T' - T_C^A$;
7: **if** $\triangle T' > \Delta T$
8: Break;
9: **else**
10: *CA* computes $t$ and $t'$ again;
11: **end if**
12: Compute $D_i' = D_i^u \cdot g^{v(t+t')}$ and $D_i'' = d_i^{\alpha} \cdot g^{-v(t+t')}$;
13: Generate $SK = \{D, D_0, D_i', D_i''\}$;
14: **return** *SK*;
15: *CA* computes $index = H(SK_{Uid}) \mod l$;
16: Store $hashchain[index] = H(SK_{Uid})$;
17: Upload $S_{Uid}$, *Uid*, and *index* to *BC*;
18: *BC* stores the mapping $(Uid, S_{Uid}) \rightarrow index$ in the IndexTable.

### D. Data Decryption

Let $\phi \subset \{1, 2, \ldots, m\}$ denote $\{i{:}\varrho(i) \in S_{Uid}\}$. If the access structure $(\mathcal{A}, \varrho)$ is satisfied by $S_{Uid}$, $\delta_i \in \mathbb{Z}_p^*$ will be calculated when $\sum_{i\in\phi} \delta_i \eta_i = s$, where $\{\eta_i\}$ is a set of valid shares of the secret parameter $S_{Uid}$. For $\forall i \in S_{Uid}$, *DU* computes:

$$\begin{aligned} DD_i &= D_i' \cdot D_i'' \\ &= \left(h_i^{sk_{Aid_i}\, ut} \cdot g^{v(t+t')}\right) \cdot \left(h_i^{\alpha t'} \cdot g^{-v(t+t')}\right) \\ &= h_i^{w}, \end{aligned} \tag{6}$$

The user proceeds to compute further, utilizing the parameter $\{\delta_i\}_{i\in\phi}$:

$$C_M = \frac{e(C_0, D)}{\prod_{i\in\phi}\left(e(C_i^1, D_0)\, e(C_i^2, DD_{\varrho(i)})\right)^{\delta_i}}$$

$$= \frac{e\left(g^s, g^\alpha \cdot g^{\beta w}\right)}{\prod_{i\in\phi}\left(e((g^\beta)^{\eta_i} \cdot h_{\varrho(i)}^{-p_i}, g^w) \cdot e(g^{p_i}, h_{\varrho(i)}^w)\right)^{\delta_i}}$$
$$= \frac{e(g,g)^{\alpha s} e(g,g)^{\beta s w}}{\prod_{i\in\phi} e(g,g)^{w\beta\eta_i\delta_i}}$$
$$= \frac{e(g,g)^{\alpha s} e(g,g)^{\beta s w}}{e(g,g)^{w\beta\sum_{i\in\phi}(\eta_i\delta_i)}}$$
$$= e(g,g)^{\alpha s}. \quad (7)$$

Next, the user proceeds to calculate the key $k$.

$$k = C'/C_M = C'/e(g,g)^{\alpha s}. \quad (8)$$

The final plaintext data $M$ can be acquired by the user by decrypting the ciphertext $CT_{(M)}$ using the symmetric key $k$.

Before decrypting the ciphertext, users can first verify the integrity of the data. Since the *CID* is a hash value generated based on the content, users can first hash the obtained ciphertext and compare the calculated hash value with the corresponding *CID*. If they are equal, it means the data has not been tampered with. Otherwise, the data may have been tampered with or damaged.

## VI. BC Verify

### A. Anomaly Detection

*DU* can query and download any encrypted data of interest from IPFS. However, before downloading the ciphertext, the key verification of the blockchain must be passed first to ensure the correctness and legality of its key. This allows the *DU* to obtain the ciphertext storage address stored in *BC*, that is, the index keyword *CID*. Keep in mind that if users' attribute set does not align with the access structure embedded in the ciphertext, they will be unable to decrypt the data, even if the ciphertext is downloaded. We utilize blockchain technology to implement the function of anomaly detection. Users can only obtain the corresponding ciphertext after the private key verification is passed. Specifically, *DU* sends a data access request to the blockchain, along with $S_{Uid}$, *Uid* and the private key $SK_{Uid}$. Upon receiving the message, the blockchain finds the corresponding *index* through the attribute set $S_{Uid}$ and *Uid* in the *Indextable*, and then sends the *index* to *CA*, requesting *CA* to return the corresponding key hash *currentHash* in the *hashchain*. The blockchain then computes $b = H(SK_{Uid})$. If $b$ = *currentHash*, the key verification succeeds, and the verification result $R$ is set to 1, otherwise $R$ is set to 0. The blockchain then returns a keyword *CID* to the user, who can use it to query and download the ciphertext from IPFS. If the verification fails, the *CID* will not be available to the user.

**Detectable Anomalies:** StorSec's hashchain-based mechanism detects several types of access-related anomalies, including:

- *Key tampering:* Use of forged keys inconsistent with the hashchain record.
- *Key reuse attacks:* Reuse of compromised keys from previous users.
- *Unauthorized access:* Key usage by users who were never issued the key by the certified authority.

**Algorithm 4** Anomaly Detection

**Input**: Attribute set $S_{Uid}$ of *Uid*, and the private key $SK_{Uid}$;
**Output**: Verification result *R*, and ciphertext keyword *CID*;
1: *DU* sends a keyword *CID* query request to the *BC*;
2: *BC* queries the *index* through $S_{Uid}$ and *Uid*;
3: Send *index* to *CA*;
4: *CA* computes $a$ = *currentHash* and returns it to *BC*;
5: *BC* computes $b = H(SK_{Uid})$ using SHA-256;
6: **if** $a \neq b$
7: Verification failed;
8: Set $R = 0$;
9: Return $(R, \perp)$;
10: **else**
11: Verification successful;
12: Set $R = 1$;
13: Retrieve and return corresponding *CID*;
14: **end if**
15: **return** $(R, CID)$;

- *Key issuance inconsistencies:* Mismatches in key issuance history traced through the hashchain.

**Handling of False Positives and False Negatives:** To reduce false positives (legitimate users mistakenly flagged), the system supports time-based tolerance and allows users to re-initiate verification via the CA. False negatives are prevented by binding all key usages to hashchain verification — every access request must match the on-chain key hash exactly, leaving no room for outdated or forged keys to pass undetected.

**System Response to Anomalies:** When an anomaly is detected, the system:

- Immediately denies access to the ciphertext.
- Logs the event on-chain, including the *Uid* and reason for failure.
- Optionally notifies the CA for further review and blacklisting of suspicious identities.

We can present the process of data acces as Algorithm 4.

### B. Auditing & Tracing

If user verification fails $(R = 0)$, we suspect issues with a *AA* or the user. The *CA* audits and tracks this. The failure may be due to a *AA* maliciously assigning incorrect attributes to the user, leading to verification failure. After the blockchain returns the failure result, it sends the user's attribute set and *Uid* to the *CA*. The *CA* tracks and confirms the suspicious *AA* that assigned and verified this attribute set for the user. Specifically, the *CA* retrieves the public key linked to the *AA* using the master key *msk* and locks the *AA* using the public key *pk*.

$$pk = \left(D_0 \cdot g^{-\alpha t'}\right)^{1/ut} = g^{sk_{Aid_i}ut/ut} = g^{sk_{Aid_i}}$$

The *CA* checks if the received attribute set matches the one assigned by the *AA*. If they don't match, it indicates that the *AA* has conducted malicious verification of the user's legitimacy. As a penalty, the identified *AA* is expelled from the system. If they match, it suggests the user obtained a key related

**Algorithm 5** Auditing of CA

**Input**: Verification result *R*;
**Output**: Responsible party;
1: **if** $R = 0$
2: *BC* sends attribute set $S_{Uid}$ to *CA*;
3: *CA* searches for corresponding *AA*;
4: Compute $pk = g^{sk_{Aid_i}}$ and retrieve assigned attribute set $S'_{Uid}$ from *Aid*;;
5: **if** $S_{Uid} \neq S'_{Uid}$
6: **return** malicious *AA* identified by *Aid*;
7: **else**
8: **return** suspicious user *Uid*;
9: **end if**
10: **else**
11: Break;
12: **end if**

to a specific attribute set through some method, attempting unauthorized access and causing key verification failure. The discovered malicious user will face severe penalties, such as being kicked out of the system.

The *AA* may have incorrectly verified the user's attributes, allowing the user to pass the blockchain verification. However, decryption fails because the user's attribute set cannot meet the access policy in the ciphertext, thus still ensuring that the data has not been illegally stolen. A minor penalty, such as a one-month suspension, can be imposed on the *AA*. We can present the process of Auditing as Algorithm 5.

## VII. Security Analysis

### A. Security Model Insurance

*1) Security and Availability:* The combination of blockchain with IPFS and data encryption enhances the security and availability of the system. The access control of identities using blockchain technology protects the CID data of IPFS from unauthorized access or tampering. The core of blockchain technology is decentralization, which maintains data through multiple nodes on the network, avoiding the risk of single point failure. This means that even if some nodes fail, the rest of the system can still operate normally, thus ensuring the availability of data, which complements the characteristics of IPFS, which is also a decentralized storage system.

*2) Data Confidentiality:* The following is the definition of this issue. Select a prime order group $\mathbb{G}_g$, where $g$ is the generator of the group, and $p$ is the prime order. We define $x, s', k_1, k_2, \ldots, k_q$ to be elements of $\mathbb{Z}_p$. An opponent is provided with $\vec{Y}$, which includes:

$$g, g^{s'}, g^{x}, \ldots, g^{x^q}, g^{x^{q+2}}, g^{x^{2q}}$$
$$\forall 1 \leq i \leq q, g^{s' \cdot k_i}, g^{x/k_i}, \ldots, g^{x^q/k_i}, g^{x^{q+2}/k_i}, \ldots, g^{x^{2q}/k_i}$$
$$\forall 1 \leq i, d \leq q, d \neq i, g^{x \cdot s' \cdot k_d/k_i}, \ldots, g^{x^q \cdot s' \cdot k_d/k_i}$$

The challenge lies in the difficulty of differentiating $e(g,g)^{x^{q+1}s'} \in \mathbb{G}_T$ from an element within $\mathbb{G}_T$ that is random.

We apply the following theorem to demonstrate the data confidentiality of StorSec:

*Theorem 1:* Assuming that the decisional q-parallel BDHE assumption is valid, there is no opponent capable of selectively breaking StorSec when faced with a challenge matrix of dimension $l^* \times n^*$, where $l^*, n^* \leq q$, using a polynomial-time algorithm.

*Proof:* When $w = sk_{Aid_i} ut + \alpha t'$ is simplified as $w$, the user's key can be refined by performing a combination operation on $D'_i$ and $D''_i$. Specifically:

$$D = g^{\alpha} g^{\beta w}, \ D_0 = g^{w}, \forall i \in S_{Uid_j}, \ DD_i = D'_i \cdot D''_i = h_i^{w},$$

where $w$ can be understood as a number that varies according to different values of $t$ and $t'$. Subsequently, the demonstration of $Theorem\ 1$ is analogous to that found in [31]. Viewers who are interested in this proof can refer to [31] for more detailed information.

Attackers cannot directly obtain a user's private key from the hashchain, as the private key is not stored in plaintext within the hashchain. The hashchain itself does not contain specific information about the private key, but generates a hash value related to the private key through hash operations. The private key of users is bound to their Global Identifier($GID$, referred to as *Uid* in this context), and each user's *GID* is unique, ensuring the uniqueness of the private key. Attackers cannot directly obtain relevant information from the private key, even in the event of an attacker stealing the private key, they cannot pass the blockchain verification to illegally steal data. ■

*3) Completeness of Data:* In this scheme, the blockchain plays the role of data index lookup and verification, while IPFS serves as the underlying protocol for actual file and data storage. After data is stored in IPFS, IPFS will return an IPFS hash, namely *CID*, which can serve as an index for accessing files and can check whether the file content has been altered. The immutability of the blockchain can ensure that this *CID* will not be tampered with.

Therefore, *DU* uses a hash function (such as SHA-256) to hash the ciphertext obtained, and compares the calculated hash value $h_{CT}$ with the hash value acquired from the blockchain, i.e., $h_C = CID$. If $h_{CT} = h_C$, it indicates that the data remains unaltered. Otherwise, there may be data tampering or damage.

### B. Resistance to Attacks

*1) DDoS Attacks:* The distributed denial of service attack is when an attacker controls a large number of infected computers (often called a "botnet") and simultaneously sends a large number of requests to the target system, causing the target system to be overloaded and unable to provide services normally. DDoS attacks usually overwhelm a single node with a large number of requests. In a decentralized distributed system based on blockchain and IPFS, data is stored in multiple independent nodes instead of being concentrated in a

TABLE II
STORAGE OVERHEAD

| | CA | AA | DO | DU |
|---|---|---|---|---|
| CP-ABE [31] | - | $2\|p\|$ | $2\|p\| + N_{CT}$ | $N_{Uid}\|p\| + \|C_u\|$ |
| AC-CP-ARBE [3] | $4\|p\| + N_u\|C_u\|$ | $(S+2)\|p\| + N_u + S$ | $2\|p\| + N_{CT}$ | $(N_{Uid}+2)\|p\| + \|C_u\| + S$ |
| Our Scheme | $(2N_A + SN_u + 4)\|p\|$ | $\|p\| + \|C_A\|$ | $2\|p\| + N_{CT}$ | $N_{Uid}\|p\| + \|C_u\|$ |

single server. It is difficult for attackers to launch attacks in a centralized manner, which increases the complexity and cost of attacks.

*2) Man-in-the-Middle Attack:* MITM attacks usually tamper with information by intercepting or forging communication data. By using blockchain technology for authentication and access control, the system can prevent unauthorized nodes or attackers from intercepting or tampering during data transmission. All transactions and interactions are encrypted, verified and recorded on multiple nodes, making it difficult for attackers to modify data unilaterally. In addition, IPFS uses content addressing and distributed storage, and data is accessed through a unique content hash value, which means that users can get data directly from the network instead of relying on a specific server, thereby reducing the risk of data being intercepted during transmission.

*3) Collusion Attack:* Since *AAs* cannot be fully trusted, there may be cases where *AAs* conspire with one another to get information from the system that they desire. A malevolent *AA* can impersonate another *AA* by using a arbitrary private key if two *AA*s conspire to create and submit keys to the *CA* for the same user concurrently. In this scenario, both $t$ and $t'$ values will be the same. In this way, colluding *AA*s can generate arbitrary keys for users, thereby illegally obtaining the privilege of *CA*. But the *CA* will verify whether $T_C^A$ is within the permitted time frame, to ensure that the same user is not reused by different *AAs* within the time interval, generating different $t$ and $t'$ values. Therefore, the situation of AA collusion attacks is strangled in the cradle. In the proposed scheme, each user is given a worldwide unique identifier *Uid*, which is connected to their key. In the key management phase, upon the generation of the user's key, a unique value $w$ is linked to each key component, and it's beyond the user's ability to compute. Therefore, multiple users cannot collude to obtain and decrypt the ciphertext.

## VIII. PERFORMANCE ANALYSIS

To ensure a meaningful and balanced evaluation of our proposed scheme, we selected the representative existing works listed in Table II as comparison baselines. These reference schemes aim to address issues related to data confidentiality, integrity, or key management, which are fundamental concerns in secure decentralized storage systems. In this section, we focus on comparing the proposed scheme with the classic CP-ABE scheme [31] and the representative AC-CP-ARBE scheme [3] in terms of key update, from other aspects such as storage overhead, communication overhead, computation efficiency and system throughput. CP-ABE scheme only involves one *AA* responsible for generating and distributing keys.

### A. Theoretical Analysis

We define some elements as follows: let $|p|$ to represent the magnitude of elements within prime order groups denoted by $p$, and let $S$ signify the attribute count. $N_u$ denotes the total user count, the average size of attributes that users possess is denoted by $N_{Uid}$, $N_A$ represents the number of *AA*s. $N_{CT}$ represents the average number of attributes in a ciphertext, $|C_u|$ represents the average size of a user certificate. while $|C_A|$ represents the average number of an *AA* certificate.

*1) Storage Overhead:* Table II presents the analysis of storage overhead for each entity. Compared with other schemes, the main overhead of *CA* in StorSec comes from storing the public keys $pk_{Aid}$ of each *AA* and maintaining a hashchain. These extra storage spaces are used for key generation, auditing, and verification. Since our scheme focuses on security, some extra storage costs are acceptable. From the perspective of *AA*, our scheme stores one more certificate than CP-ABE [31], because StorSec explicitly considers the distribution of *AA* certificates when *AA* registers with *CA*. AC-CP-ARBE [3] transfers most of the work of *CA* to *AA*, and also maintains a user list. In AC-CP-ARBE scheme, the storage overhead of every user is relatively large, mainly because the user needs to store the attribute certificate (AC) for key update.

*2) Communication Overhead:* Table III shows the communication costs of different entities in different phases of StorSec, CP-ABE [31] and AC-CP-ARBE [3]. In the initial setup stage, schemes involving multiple authorities incur extra communication costs within the *CA*, because the registration of *AA*s is a necessary sacrifice. To ensure the auditability of *AA*s, additional overhead is incurred during the StorSec key generation and distribution phase. Our StorSec is equally efficient as Waters' schemes in the encryption and decryption phase.

### B. Experimental Evaluation

We assess the efficiency of our model by implementing the scheme according to the structure proposed by [32], the algorithm is deployed on Apache server through IntelliJ IDEA, and we apply the Java pairing-based cryptography library (IPBC). The experiment was completed using the Windows 11 operating system, powered by an Intel Core i5-12500H processor and equipped with 16G memory. The security parameters are set such that $p$ is 128 bits and $q$ is 256 bits. The physical experiment environment requires three hosts and a router. Three local area network (LANs) were constructed by connecting the physical hosts through the router to simulate different areas where groups of IoT devices and service providers are located. Network nodes were deployed using virtual machine-based hosts to form clusters of nodes with different IP addresses to simulate the P2P network

TABLE III
COMMUNICATION OVERHEAD

| | System Initialization | | | Key Generation | | | Encrypt/Decrypt | Audit & Trace |
|---|---|---|---|---|---|---|---|---|
| | $CA$ | $AA$ | User | $CA$ | $AA$ | DU | DO/DU | $CA$ |
| CP-ABE [31] | - | $N_u\|C_u\|$ | $\|C_u\|$ | - | $(N_{Uid}+2)\|p\|+\|C_u\|$ | $(N_{Uid}+2)\|p\|+\|C_u\|$ | $(N_{CT}+2)\|p\|$ | - |
| AC-CP-ARBE [3] | $(N_u+N_A)\|p\|+N_u\|C_u\|+N_A\|C_A\|$ | $4\|p\|+\|C_u\|$ | $\|p\|+\|C_u\|$ | $(N_{Uid}+4)\|p\|$ | $(N_{Uid}+4)\|p\|+\|C_u\|$ | $(N_{Uid}+4)\|p\|+\|C_u\|$ | $(N_{CT}+2)\|p\|$ | - |
| Our Scheme | $(N_u+N_A)\|p\|+N_u\|C_u\|+N_A\|C_A\|$ | $\|p\|+\|C_A\|$ | $\|p\|+\|C_u\|$ | $(4N_{Uid}+5)\|p\|+\|T_C^A\|$ | $(2N_{Uid}+4)\|p\|+2\|T_C^A\|+\|C_u\|$ | $(N_{Uid}+2)\|p\|+\|C_u\|$ | $(N_{CT}+2)\|p\|$ | $2\|p\|$ |

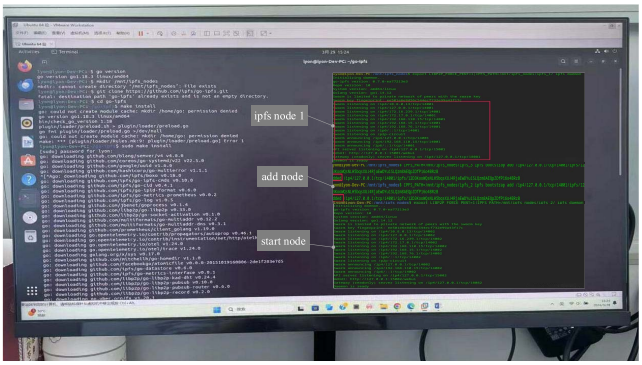


Fig. 7. An illustration of IPFS deployment.

transmission structure of the blockchain network. The main role of the blockchain in this scheme is for verification, and the choice of its consensus mechanism has little impact on the overall scheme performance. IPFS private network deployment is shown in Fig. 7.

*1) Storage Overhead:* We compared the proposed storage scheme with the AC-CP-ARBE scheme, which runs in the cloud environment. Fig. 8 shows the time efficiency of IPFS and cloud storage. The experiment selected different data sizes and recorded the upload, download and concurrent test time of cloud storage and IPFS. As shown in Fig. 8(a)(b), the upload and download times of IPFS and cloud storage increase with the size of the data. However, the increase in IPFS is smaller, and its advantage becomes more obvious as the data volume increases. To further test system performance, under unchanged conditions, multiple identical files were uploaded simultaneously to test the impact of concurrency on file transfer. The experimental results of uploading files of different data sizes 10 times in a row are shown in Fig. 8(c). Since IPFS is a content-addressable distributed file system, the content of the files it stores will not be duplicated, and only when the file is uploaded for the first time will the file upload time consumption be larger. When the same data file is uploaded multiple times, the cloud server will store it repeatedly. Therefore, the cloud upload time is much greater than IPFS.

*2) Computation Efficiency:* We compare our scheme with AC-CP-ARBE [3] and CP-ABE [31] in system initialization. Although the construction of the hashchain results in our scheme taking more time, it enhances its security. Therefore, we think that this extra overhead is acceptable, as can be seen from Fig. 9(a). During the phase of generating keys, as shown in Fig. 9(b), we compare StorSec with EHRChain [33] and Waters' scheme. StorSec is faster and more efficient because of multiple *AA*s. During the phase of encryption and decryption, as depicted in Fig. 9(c),(d), our StorSec is as effective as Waters' scheme. The cost of retrieving the key or ciphertext is displayed in Fig. 9(e). We use the instance simulated by scheme BC-SABE [2] to analyze the effectiveness of the search algorithm. It is observable that the time expense of the BC-SABE scheme's search algorithm correlates linearly with the attribute count, while our StorSec scheme uses the advantage of the hash table to provide a fast retrieval mechanism, reducing the time overhead of retrieval to O(1). Key update phase in CP-ABE involves retrieval, update, and distribution. Traditional methods need to find and regenerate all users' keys with revoked attributes. This is more complex than key generation. Fig. 9(f) shows the key update time of StorSec and other schemes. StorSec is faster than AC-CP-ARBE, EHRchain and CP-ABE when revoking attributes. In the key update test scenario, one attribute is revoked from five attributes. StorSec has an efficient retrieval mechanism that saves time for finding users. AC-CP-ARBE uses a third-party authority for key update, but *AA* still needs to find and notify users, and send them the attribute revocation configuration. EHRchain reduces the key size by using a semi-hidden strategy, but does not improve the retrieval time.

*3) Average Transaction Delay:* In the experiment, cloud architecture and IPFS are deployed at the bottom of each node. Fig. 10 shows the average delay of traditional cloud architecture and IPFS architecture under different transaction numbers when the number of nodes $n$ is 6, 9 and 12 respectively. It can be seen that with the growth of transaction number, the mean transaction delay of both architectures shows an upward trend. The upper limit of cloud-based architecture Fig. 10(a) is about 2.5$s$, while the upper limit of IPFS-based distributed architecture Fig. 10(b) is about 2$s$. Compared with cloud architecture, the average transaction delay of IPFS architecture is smaller.

*4) System Throughput Measurement:* To further evaluate our system's performance, we conduct a comparison of the system's average throughput under identical operations. Fig. 11 compares the average throughput of traditional cloud

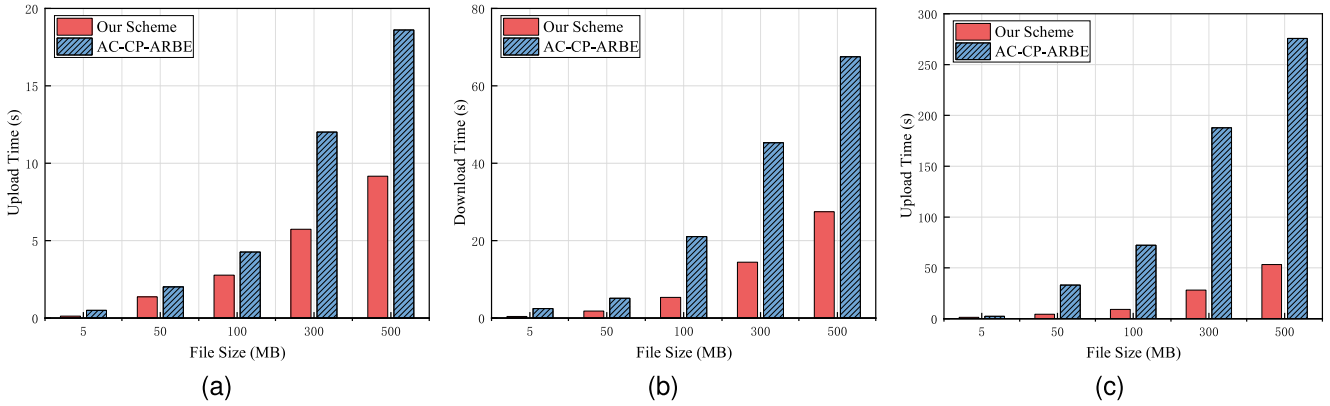


Fig. 8. Time efficiency of IPFS and cloud storage systems.

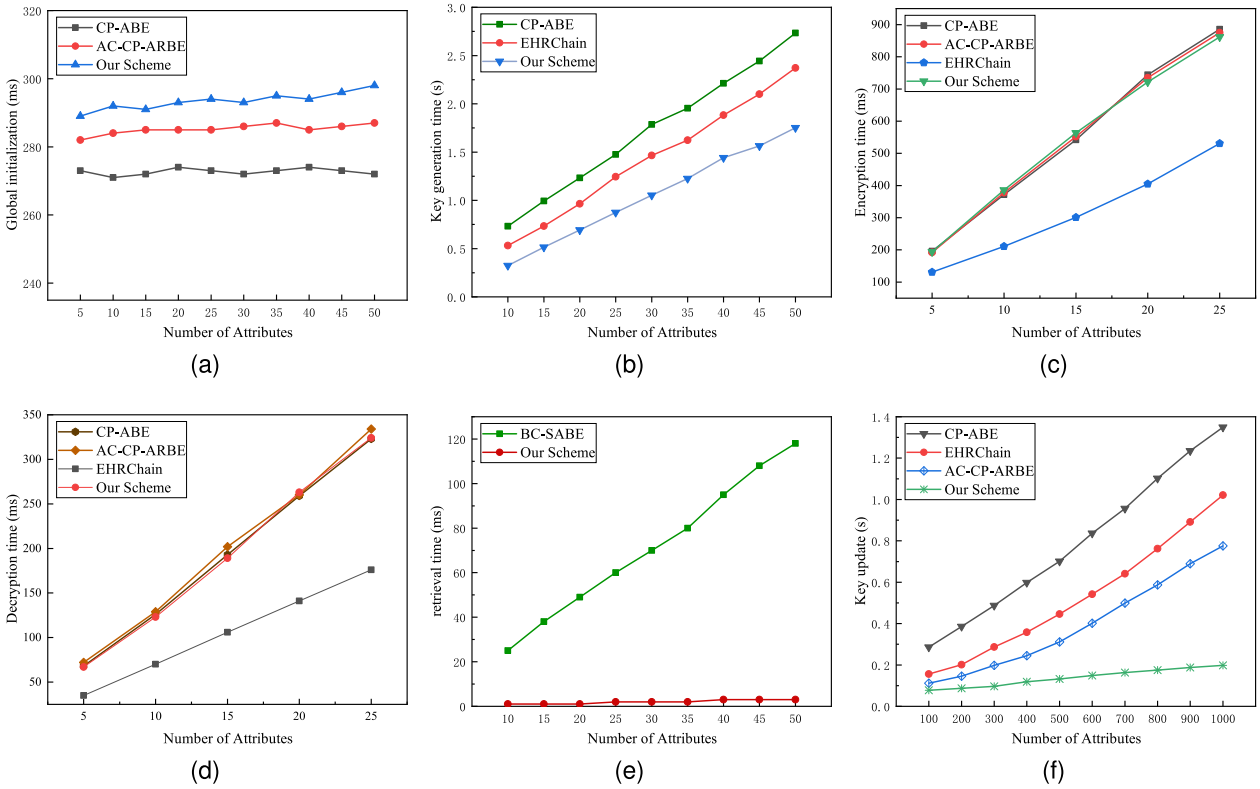


Fig. 9. (a) System global initialization time. (b) Key generation time. (c) Plaintext encryption time. (d) Ciphertext decryption time. (e) Retrieval execution time. (f) Key update time.

and IPFS architectures with different transactions and nodes. With a constant number of transactions, an increase in the number of nodes set in the architecture leads to a larger average throughput. Compared with traditional cloud architecture, IPFS architecture has higher average throughput because IPFS uses distributed, content-addressed, and peer-to-peer design, which reduces network distance, time, bandwidth, and latency. IPFS-based storage system can process transactions faster and improve transaction efficiency.

## IX. Conclusion

In this article, we explore the issue of data security storage in the IoT environment. Different from prior works, our scheme grants data to be securely and efficiently accessed by those users who have been authenticated. To achieve secure and efficient storage and access to IoT data, we employ a storage system that combines blockchain and IPFS. We design a new heterogeneous framework and adopt improved attribute encryption algorithms to eliminate single-point performance bottlenecks. This not only realizes fine-grained access control and ensures the security of the data during transmission but also improves the performance of key generation. To prevent data from being illegally accessed, we design a blockchain key verification method to monitor the improper behavior of the responsible party. Moreover, our scheme is safe and efficient, as demonstrated by our detailed security and performance analysis, and it can withstand collusion attacks

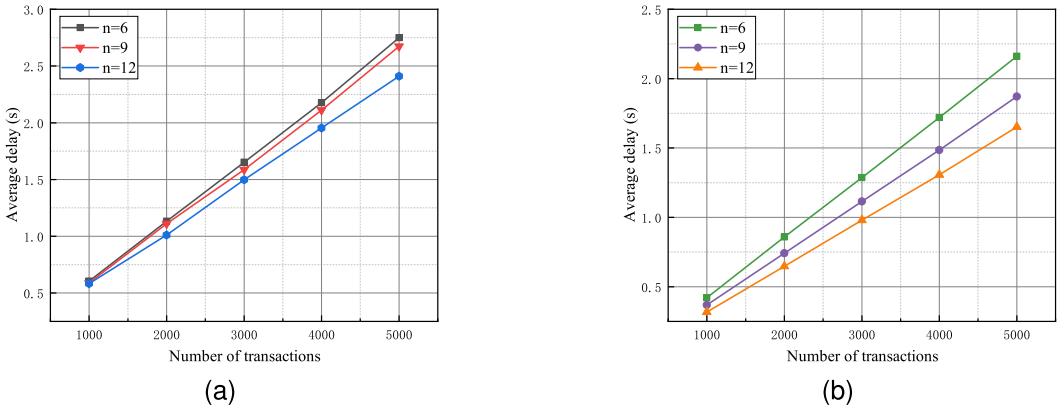


Fig. 10. Average latency of traditional cloud architecture (a) and IPFS based architecture (b) for different number of transactions when the number of nodes $n$ is 6, 9 and 12 respectively.

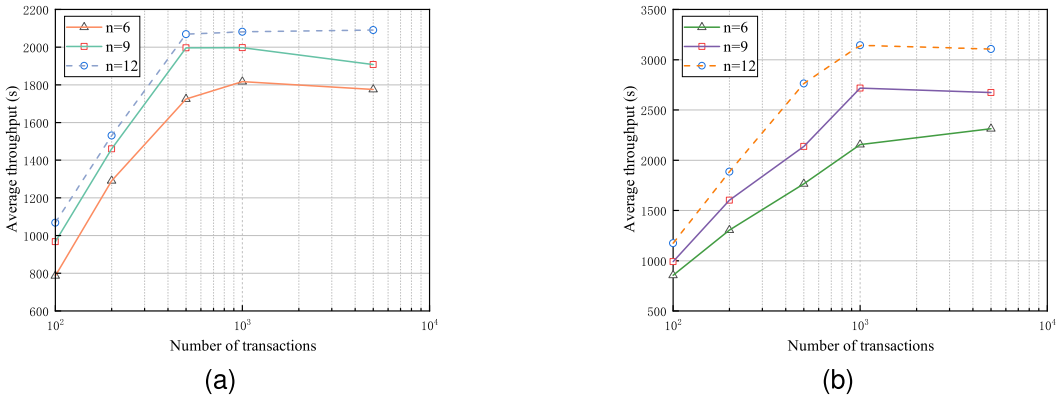


Fig. 11. The traditional cloud architecture (a) and the IPFS-based architecture (b) show the trend of system throughput changes under varying quantities of transactions when the number of nodes $n$ is 6, 9, and 12 respectively.

from institutions or users effectively, as well as attacks from malicious users and colluding institutions. Although StorSec achieves secure and auditable access control over decentralized storage, it still has some limitations. First, due to the static nature of CP-ABE, the current design does not support fine-grained or real-time revocation of keys and attributes. While outdated keys can be detected through the hashchain-based verification mechanism, there is currently no efficient method for immediate revocation, which may impact the integrity of access control and auditing in dynamic environments. Second, in distributed storage systems like IPFS, data availability is inherently tied to the robustness of the incentive mechanism. If a large number of storage nodes go offline, users may be unable to retrieve their data, threatening system reliability.

To address these challenges, in the future, we plan to enhance StorSec by integrating time-constrained ABE schemes (e.g., T-ABE) and attribute expiration timestamps to support dynamic and low-overhead revocation. In addition, we will incorporate the Filecoin incentive layer to promote long-term node availability and stable bandwidth provisioning within the IPFS network. This will improve data resilience and ensure continuous access in decentralized environments. We will also explore advanced key management techniques, including periodic key refreshing and revocation handling, to strengthen the system's security and scalability.

## REFERENCES

[1] S. Chen. "Digital wisdom point of view: Multi-directional empowerment of China's artificial intelligence arithmetic development to run out of 'acceleration." 2022. [Online]. Available: https://m.thepaper.cn/baijiahao_18876314/

[2] S. Liu, J. Yu, Y. Xiao, Z. Wan, S. Wang, and B. Yan, "BC-SABE: Blockchain-aided searchable attribute-based encryption for cloud-IoT," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 7851–7867, Sep. 2020.

[3] S. Fugkeaw, "Secure data sharing with efficient key update for industrial cloud-based access control," *IEEE Trans. Services Comput.*, vol. 16, no. 1, pp. 575–587, Jan./Feb. 2023.

[4] S. Namasudra, "Fast and secure data accessing by using DNA computing for the cloud environment," *IEEE Trans. Services Comput.*, vol. 15, no. 4, pp. 2289–2300, Jul./Aug. 2020.

[5] L. Zhang, C. Jin, H.-p. Huang, X. Fu, and R.-c. Wang, "A trajectory privacy preserving scheme in the CANNQ service for IoT," *Sensors*, vol. 19, no. 9, p. 2190, 2019.

[6] S. Zhang, J. He, W. Liang, and K. Li, "MMDS: A secure and verifiable multimedia data search scheme for cloud-assisted edge computing," *Future Gener. Comput. Syst.*, vol. 151, pp. 32–44, Feb. 2024.

[7] S. Zhang, B. Hu, W. Liang, K.-C. Li, and B. B. Gupta, "A caching-based dual K-anonymous location privacy-preserving scheme for edge computing," *IEEE Internet Things J.*, vol. 10, no. 11, pp. 9768–9781, Jun. 2023.

[8] A. Kokolis, A. Psistakis, B. Reidys, J. Huang, and J. Torrellas, "Distributed data persistency," *IEEE Micro*, vol. 42, no. 4, pp. 107–115, Jul./Aug. 2022.

[9] H. Tian, X. Ge, J. Wang, C. Li, and H. Pan, "Research on distributed blockchain-based privacy-preserving and data security framework in IoT," *IET Commun.*, vol. 14, no. 13, pp. 2038–2047, 2020.

[10] C. Liu, "HPCLS-BC: A novel blockchain framework using heterogeneous peer-node and cloud-based ledger storage for Internet of Things applications," *Future Gener. Comput. Syst.*, vol. 150, pp. 364–379, Jan. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X23003473
[11] E. Daniel and F. Tschorsch, "IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 31–52, 1st Quart., 2022.
[12] L. Cao, X. Jiang, Y. Zhao, S. Wang, D. You, and X. Xu, "A survey of network attacks on cyber-physical systems," *IEEE Access*, vol. 8, pp. 44219–44227, 2020.
[13] S. Rana and D. Mishra, "An authenticated access control framework for digital right management system," *Multimedia Tools Appl.*, vol. 80, pp. 25255–25270, Apr. 2021.
[14] A. K. Singh, K. Acharya, and R. Dutta, "Cloud assisted semi-static secure accountable authority identity-based broadcast encryption featuring public traceability without random oracles," *Ann. Telecommun.*, vol. 78, nos. 1–2, pp. 79–90, 2023.
[15] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. 24th Annu. Int. Conf. Theory Appl. Cryptogr. Tech. Adv. Cryptol. (EUROCRYPT)*, 2005, pp. 457–473.
[16] H. Deng et al., "Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts," *Inf. Sci.*, vol. 275, pp. 370–384, Aug. 2014.
[17] L. Zhang, G. Hu, Y. Mu, and F. Rezaeibagha, "Hidden ciphertext policy attribute-based encryption with fast decryption for personal health record system," *IEEE Access*, vol. 7, pp. 33202–33213, 2019.
[18] S. Yao, R. V. J. Dayot, I.-H. Ra, L. Xu, Z. Mei, and J. Shi, "An identity-based proxy re-encryption scheme with single-hop conditional delegation and multi-hop ciphertext evolution for secure cloud data sharing," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 3833–3848, 2023.
[19] M. Chase, "Multi-authority attribute based encryption," in *Proc. 4th Theory Cryptogr. Conf. (TCC)*, 2007, pp. 515–534.
[20] K. Xue et al., "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 953–967, 2017.
[21] S. Das and S. Namasudra, "Multiauthority CP-ABE-based access control model for IoT-enabled Healthcare infrastructure," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 821–829, Jan. 2023.
[22] J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Proc. 12th Int. Conf. Inf. Secur. (ISC)*, 2009, pp. 347–362.
[23] J. Li, Q. Huang, X. Chen, S. S. Chow, D. S. Wong, and D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability," in *Proc. 6th ACM Symp. Inf., Comput. Commun. Secur.*, 2011, pp. 386–390.
[24] J. Li, K. Ren, and K. Kim, "A2BE: Accountable attribute-based encryption for abuse free access control," Cryptol. ePrint Arch., IACR, Bellevue, WA, USA, Rep. 2009/118, 2009. [Online]. Available: https://eprint.iacr.org/2009/118
[25] J. Chen, Y. Wang, Z. Huang, C. Ruan, and C. Hu, "A decentralized public auditing scheme for secure cloud storage based on blockchain," *Wireless Commun. Mobile Comput.*, vol. 2022, no. 1, 2022, Art. no. 3688164.
[26] J. Lu, J. Shen, P. Vijayakumar, and B. B. Gupta, "Blockchain-based secure data storage protocol for sensors in the industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5422–5431, Aug. 2021.
[27] K. Azbeg, O. Ouchetto, and S. J. Andaloussi, "BlockMedCare: A healthcare system based on IoT, blockchain and IPFS for data management security," *Egyptian Inform. J.*, vol. 23, no. 2, pp. 329–343, 2022.
[28] Z. Ullah, B. Raza, H. Shah, S. Khan, and A. Waheed, "Towards blockchain-based secure storage and trusted data sharing scheme for IoT environment," *IEEE Access*, vol. 10, pp. 36978–36994, 2022.
[29] W. Yang, Z. Guan, L. Wu, X. Du, and M. Guizani, "Secure data access control with fair accountability in smart grid data sharing: An edge blockchain approach," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8632–8643, May 2021.
[30] L. Bader et al., "Blockchain-based privacy preservation for supply chains supporting lightweight multi-hop information accountability," *Inf. Process. Manage.*, vol. 58, no. 3, 2021, Art. no. 102529.
[31] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. 14th Int. Workshop Public Key Cryptogr.*, 2011, pp. 53–70.
[32] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy (SP)*, 2007, pp. 321–334.
[33] F. Li, K. Liu, L. Zhang, S. Huang, and Q. Wu, "EHRChain: A blockchain-based EHR system using attribute-based and homomorphic cryptosystem," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2755–2765, Sep./Oct. 2022.

**Shiwen Zhang** received the Doctorate degree in computer science and electronic engineering from Hunan University in 2016. He currently holds an Associate Professorship with the School of Computer Science and Engineering, Hunan University of Science and Technology. As an IEEE and CCF affiliate, his scholarly pursuits encompass identity verification, safeguarding privacy within wireless body area networks (WBAN), as well as advancing cloud computing, data protection, and cybersecurity measures.

**Wen Zhang** received the B.S. degree from Zhoukou Normal University. She is currently pursuing the Master of Science degree in computer science and engineering with the Hunan University of Science and Technology. She delves into the realm of blockchain technology, focusing on its security aspects for storage solutions.

**Wei Liang** (Senior Member, IEEE) received the Ph.D. degree in computer science and technology from Hunan University in 2013. He currently serves as a Professor and the Dean with the School of Computer Science and Engineering, Hunan University of Science and Technology. His academic focus lies in the areas of identity verification within WBAN and the security protocols of wireless sensor networks.

**Wenqiang Jin** (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Arlington, TX, USA, in 2021. He is currently a Professor with the College of Computer Science and Electronic Engineering, Hunan University, China. His research interests include mobile security, the loT security, and ubiquitous computing.

**Keqin Li** (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, Beijing, in 1985, and the Ph.D. degree from the University of Houston in 1990. He is renowned as a SUNY Distinguished Professor with the State University of New York in Albany and holds the title of National Distinguished Professor with Hunan University, Changsha. With over 850 publications to his name, including journal articles, book chapters, and conference papers, his expertise spans parallel and distributed computing, as well as security in cloud computing. He serves as an Associate Editor for *ACM Computing Surveys* and *CCF Transactions on High Performance Computing*, and has been a part of the editorial boards for several IEEE journals. He is also recognized as a Fellow of the AAIA and belongs to the *Academia Europaea*.