Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.054372

ARTICLE

Research on IPFS Image Copyright Protection Method Based on Blockchain

Xin Cong, Lanjin Feng* and Lingling Zi

College of Computer and Information Science, Chongqing Normal University, Chongqing, 401331, China

*Corresponding Author: Lanjin Feng. Email: 2022210516044@stu.cqnu.edu.cn

Received: 26 May 2024 Accepted: 20 August 2024 Published: 15 October 2024

ABSTRACT

In the digital information age, distributed file storage technologies like the InterPlanetary File System (IPFS) have gained considerable traction as a means of storing and disseminating media content. Despite the advantages of decentralized storage, the proliferation of decentralized technologies has highlighted the need to address the issue of file ownership. The aim of this paper is to address the critical issues of source verification and digital copyright protection for IPFS image files. To this end, an innovative approach is proposed that integrates blockchain, digital signature, and blind watermarking. Blockchain technology functions as a decentralized and tamper-resistant ledger, recording and verifying the source information of files, thereby establishing credible evidence of file origin. A digital signature serves to authenticate the identity and integrity of the individual responsible for uploading the file, ensuring data security. Furthermore, blind watermarking is employed to embed invisible information within images, thereby safeguarding digital copyrights and enabling file traceability. To further optimize the efficiency of file retrieval within IPFS, a dual-layer Distributed Hash Table (DHT) indexing structure is proposed. This structure divides file index information into a global index layer and a local index layer, significantly reducing retrieval time and network overhead. The feasibility of the proposed approach is demonstrated through practical examples, providing an effective solution to the copyright protection issues associated with IPFS image files.

KEYWORDS

Blockchain; copyright protection; IPFS; distributed hash table; digital signature

1 Introduction

The digital age has seen an explosive growth of information, driven by the widespread adoption of technologies like big data, the Internet of Things (IoT), and cloud computing. In this context, distributed file storage systems have emerged as one of the key technologies for handling massive data. These systems segment large files into smaller chunks, storing them across multiple network nodes, thus facilitating efficient file transmission and sharing. The InterPlanetary File System (IPFS) [1], a representative distributed file storage system, addresses a series of issues inherent in traditional centralized storage systems, such as poor transparency and low security, through decentralization and content addressing. However, with the widespread adoption of IPFS, determining file ownership has become a significant issue [2]. Although IPFS locates files using hash values, it lacks a mechanism to ensure file authenticity and ownership, posing risks for file theft, misinformation, and copyright

disputes. Additionally, IPFS divides the file into multiple data chunks, storing them on different nodes based on the Merkle Directed Acyclic Graph (Merkle DAG) structure. When retrieving a file, it must fetch data chunks from multiple nodes via the Distributed Hash Table (DHT), leading to increased query time and network latency. This limitation poses challenges to file retrieval efficiency, resulting in slower file access for users.

This study aims to address the challenges associated with copyright protection for IPFS image files and to enhance the efficiency of file retrieval. By leveraging the technical features of blockchain, combined with digital signature [3] and blind watermarking [4], a comprehensive solution is constructed to ensure the genuine ownership and copyright traceability of IPFS image files. Furthermore, a dual-layer DHT indexing structure is proposed to enhance file retrieval efficiency. Traditional double-layer indexing structures typically group the Content Identifiers (CIDs) of similar files into an index file, which is then upload this file to IPFS to obtain its CID, forming a double-layer index. In contrast, the dual-layer DHT indexing proposed in this study is based on a DHT design, comprising a global DHT and a local DHT. In this structure, the global index layer is responsible for storing mappings of file hash to root node position, while the local index layer stores mappings of sub-chunk hash to chunk node position. This layered indexing effectively reduces the number of queries in the Distributed Hash Table, thereby making file retrieval and acquisition more efficient.

The contributions and main works of this paper are summarized as follows:

(1) To improve file retrieval efficiency in IPFS, the traditional distributed hash table structure is enhanced by introducing an additional local index layer. This modification speeds up data chunk retrieval and location, resulting in a 33% increase in retrieval efficiency.

(2) A blockchain-based framework for file source verification is implemented. The blockchain is used to record source information and provide credible transaction records for verifying file origins. Smart contract methods are designed to verify the identity of file uploaders and ensure transparency of file ownership.

(3) A solution for file copyright protection and traceability is devised by integrating blockchain, digital signature, and blind watermarking. The solution ensures the authenticity of the file uploader's identity and confirms digital copyright.

The rest of the paper is organized as follows: Related work materials are discussed in Section 2. The implementation method of the dual-layer DHT indexing structure is described in Section 3. In Section 4, a blockchain-based IPFS image file copyright protection scheme is designed. In Section 5, the dual-layer DHT indexing structure method and copyright protection scheme are evaluated and compared with existing methods. Finally, conclusions are drawn in Section 6.

2 Related Work

2.1 IPFS

IPFS is a distributed file storage network made up of multiple autonomous nodes. When a file is uploaded to IPFS, it is partitioned into chunks, with each chunk assigned a unique identifier called a Context Identifier (CID). These CIDs are organized into a Merkle DAG, where each node contains the hash of one or more chunks and a link to its child nodes [5]. After dividing a file into chunks and forming a Merkle DAG, it registers itself as a provider of chunks through the DHT. Node discovery, connection, and data transfer are all managed through DHT. This allows IPFS nodes to request providers through DHT and connect with them to retrieve files [6].

The DHT of IPFS employs the Kademlia algorithm [7] for network routing and location requests. When a node joins the IPFS network, it is assigned a 160-bit identifier called *NodeId*. The distance d between two nodes is defined as the Exclusive OR (XOR) of the two *NodeId* ($\oplus$ is the XOR operation), denoted as:

$$d\left(\text{Node}_{1},\ \text{Node}_{2}\right) = NodeID_1 \oplus NodeID_2 \tag{1}$$

Each node locally maintains a routing table, which consists of 160 K-buckets. Each K-bucket contains K routing entries of other nodes, represented in the form of $< NodeId, address >$. When discovering a new node, the local node $Node_{local}$ places it in different K-buckets based on the distance between the new node and itself. In the ith K-bucket, nodes within the range $[2^i, 2^i + 1)$ are stored. When a node $Node_{local}$ receives a lookup message for a target identifier $NodeId_{target}$ from a target node $Node_{target}$, it calculates the distance between itself and the target node, $d\left(Node_{local}, Node_{target}\right)$, and forwards the message to the α closest nodes in the corresponding K-bucket, where α is the message parallelism. According to the construction rule of K-buckets, the next hop node $Node_{next}$ and the target node $Node_{target}$ satisfy $d\left(Node_{next}, Node_{target}\right) \leq d\left(Node_{local}, Node_{target}\right)/2$, ensuring that the message can be forwarded from any node to the target node within $logN$ hops. Additionally, increasing the message parallelism α can increase the number of neighbor nodes selected when nodes transmit messages, thereby improving the success rate and efficiency of queries.

2.2 Blockchain-Based Image Copyright Protection

Blockchain technology is widely applied to digital content copyright protection due to its decentralization, tamper-resistance, traceability, and other advantages [8,9]. Ning et al. [10] implemented an image transaction system using blockchain and IPFS technology, ensuring data accessibility. Muwafaq et al. [11] designed an image copyright management framework combining blockchain and perceptual hash technology to achieve third-party-free distribution of copyrighted works, protecting copyright owners. Kumar et al. [12] proposed a copyright protection system based on IPFS and blockchain to enhance the security of industrial image and video data. Zhang et al. [13] proposed a digital image copyright protection method based on blockchain and zero-trust mechanism, ensuring that only authorized users can access and use the image, thereby protecting its copyright. Meng et al. [14] proposed a copyright management method combining digital watermarking, blockchain, perceptual hashing, and other technologies to enhance the effectiveness of digital watermarking in copyright protection. Wang et al. [15] proposed a secure image copyright protection framework based on blockchain, which solves the data expansion problem of blockchain through IPFS, and improves the zero-watermark algorithm to realize the copyright traceability of images. Chen et al. [16] proposed a zero-watermark image protection framework based on blockchain, which adopts IPFS to solve the problem of efficient storage and sharing of large files on blockchain, and realizes the management and transaction of copyright information through smart contracts.

Although a variety of technologies and methods have emerged for digital content copyright protection, existing research has mainly targeted the traditional Internet environment. The issues of image file ownership and copyright protection in distributed file systems such as IPFS have not been adequately addressed. In this paper, we introduce the digital signature technique [17], which generates and verifies the file uploader's digital signature, using a key pair linked to the user's identity. This process ensures the file's integrity and verifies the uploader's identity.

2.3 IPFS Performance Optimization

As a distributed file system, IPFS faces significant challenges in terms of file retrieval efficiency [18]. Researchers in the field of distributed systems have been continuously conducting diverse studies to optimize data storage and retrieval performance. Zhu et al. [19] applied distributed B (balance)+ Tree and HashMap structures to distributed storage systems to enable keyword search. The B+ Tree structure optimizes disk access, making it suitable for block access in decentralized storage systems. The HashMap, optimized for fast lookups, complements the B+ Tree by efficiently handling index data. They also implemented an index merging algorithm to lower network costs and reduce response time. However, with the increase of data volume and network traffic, distributed B+ Tree and HashMap will cause network congestion. Cao et al. [20] proposed a double-layer inverted indexing structure to enhance keyword search efficiency and accuracy. The first layer offers a high-level overview of data locations, while the second layer contains detailed information about the data chunks. This hierarchical structure quickly narrows the search area before detailed searching, optimizing both search speed and accuracy. While effective for smaller datasets, the double-layer structure may face scalability issues as the size of the data grows. Khudhur et al. [21] designed a decentralized search engine, the core idea of which is to store the association between keywords and CID in the DHT used by IPFS. They employed result caching to speed up the processing of partially repeated queries, and a variant of the Bloom filter to quickly check the availability of the cache, thereby reducing bandwidth usage and query latency. While caching improves performance, it may be less effective for entirely new or infrequently accessed queries, potentially limiting its effectiveness in dynamic environments. Yao et al. [22] constructed a peer-to-peer cross-cluster data query system based on IPFS, which includes theoretical modeling of DHT query latency and introduces a Fat Merkle Tree structure. They designed a DHT caching mechanism to reduce query latency, aiming to enhance the cross-cluster data retrieval performance. However, the theoretical modeling and Fat Merkle Tree structure add complexity, potentially making the system harder to implement and maintain.

To overcome the limitations of existing approaches, this paper proposes a dual-layer DHT indexing structure. This structure provides basic location information for files within the global DHT and stores detailed index information for file chunks in the local DHT, thus enhancing the efficiency of IPFS file retrieval.

3 Dual-Layer DHT Indexing Structure Design

In IPFS, all file chunks are organized into a Merkle DAG. The root hash and its corresponding child hashes are stored in IPFS nodes that can be located through the DHT. The leaf hashes of the Merkle DAG represent the hash values of the individual data chunk. Suppose a node (referred to as the requesting node) needs to access a specific file in IPFS, it calculates the distance between the root hash and its node hash through the DHT to locate the IPFS node that stores the root hash information. As shown in Fig. 1, the requesting node obtains the child hash information from the root hash, and then retrieves the leaf hash information. Finally queries the DHT based on the leaf hash to find the specific location of each chunk and reconstruct the entire file. This indicates that for large files, IPFS requires multiple DHT accesses to obtain the complete file. The key effort in the design of the dual-layer DHT indexing structure is to reduce DHT accesses to improve the efficiency of IPFS network queries.

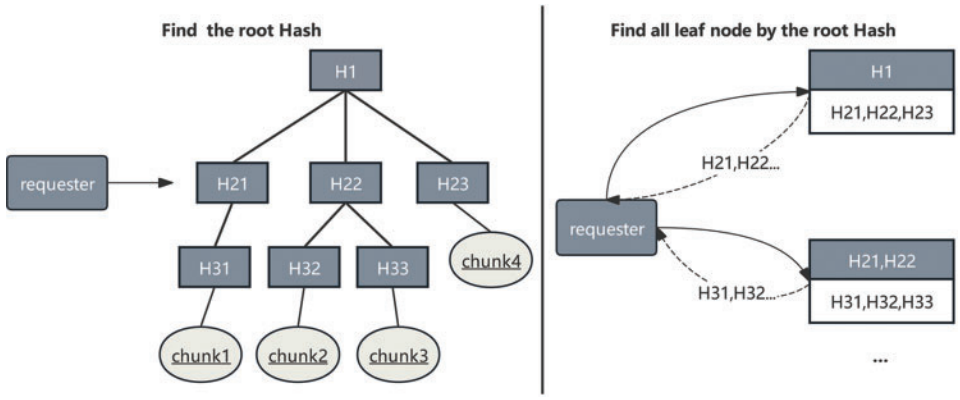


Figure 1: IPFS gets the hash of the data chunk

3.1 Scheme Design

This paper introduces an additional index layer, dividing the DHT structure into a global DHT and a local DHT. The global DHT functions similarly to the traditional DHT, storing the mapping between file CID and provider nodes, while the local DHT only stores the mapping between chunk CIDs and their corresponding location information. The file retrieval process is as follows:

(1) Obtaining chunk CIDs: A node needing to retrieve a file first queries the global DHT for the provider node corresponding to the file CID, then obtains the Merkle DAG structure of the file and its chunk CIDs from the provider node, as shown in Fig. 2.

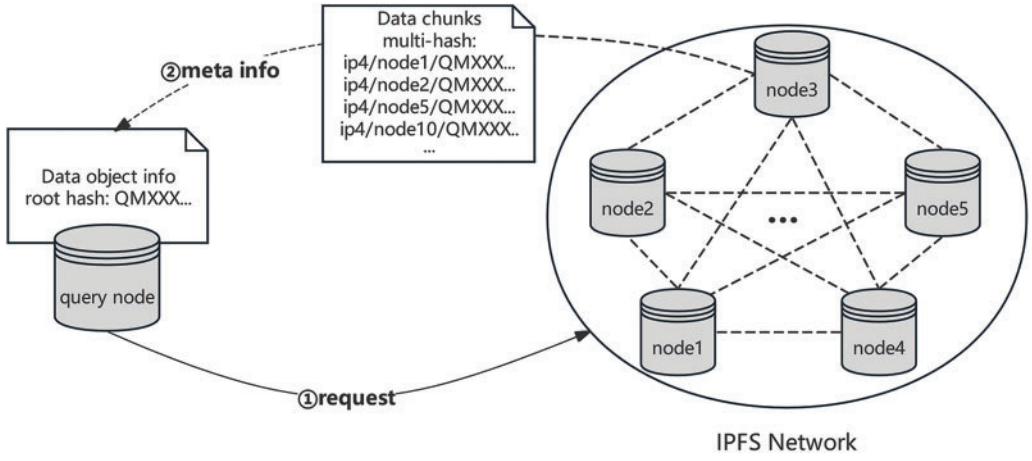


Figure 2: Global DHT gets the chunk hash

(2) Chunk location query: The node uses the obtained chunk CIDs to query the local DHT where the chunk CIDs are stored. Since the local DHT contains the mapping information between chunk CIDs and their storage locations, it directly returns the location information (IP (Internet Protocol) address) of the chunks.

(3) Data retrieval: Once the node has obtained the storage locations of the data chunks, it can quickly locate the nodes storing these chunks and download them. After all data chunks are acquired, the file is reconstructed according to the Merkle DAG structure, as shown in Fig. 3.

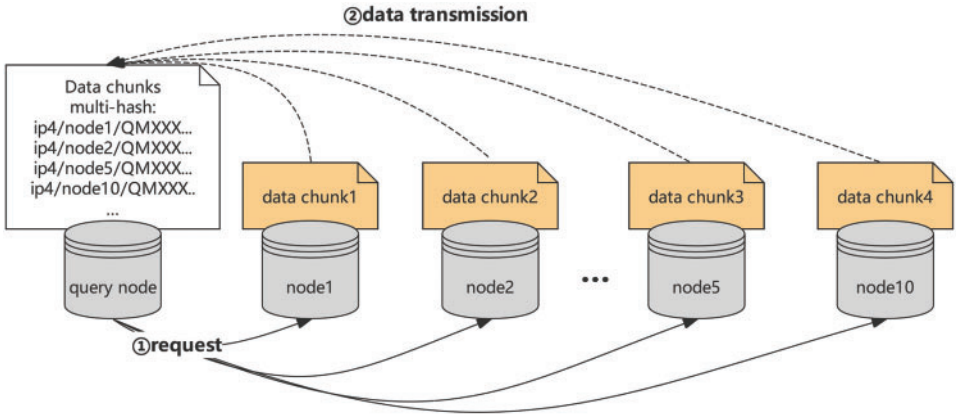


Figure 3: Local DHT obtains data chunk content

The overall architecture of the dual-layer DHT indexing is shown in Fig. 4. Its main advantage is the reduction in the number of DHT queries required for file retrieval. The node can directly locate and download data chunks without the need to iteratively query intermediate nodes of the Merkle DAG, thus reducing the overall size of the DHT. Consequently, it improves the efficiency and reliability of file queries by decreasing the number of hops and associated delays in the DHT network.

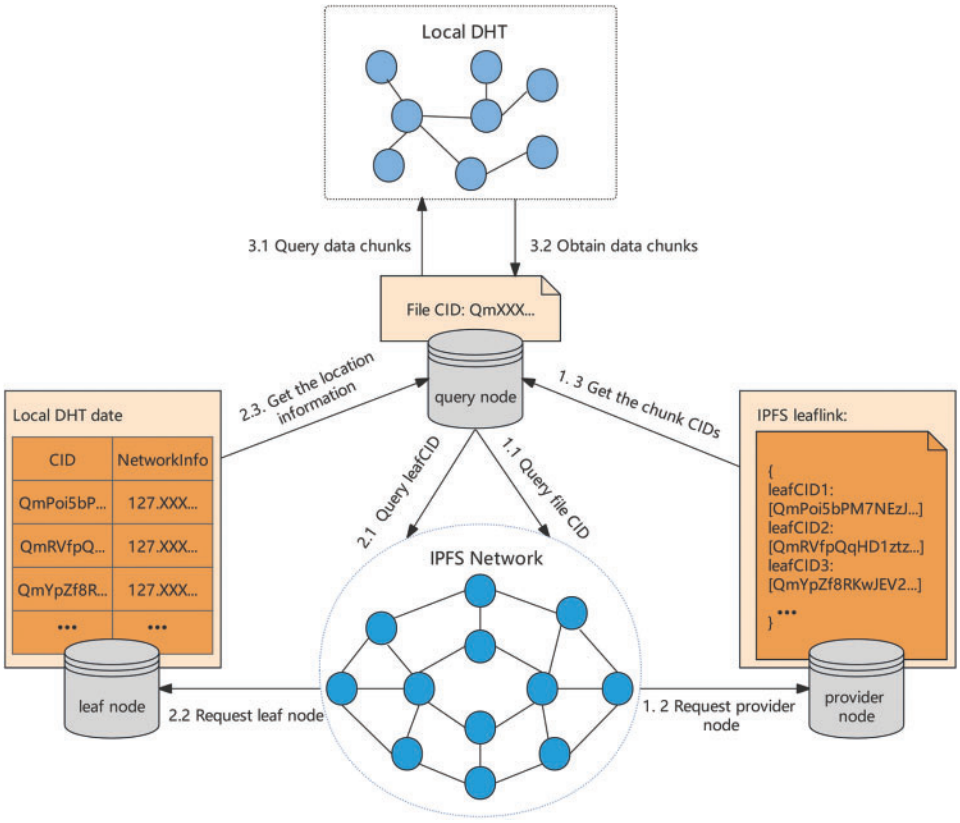


Figure 4: Overall architecture of the dual-layer DHT indexing

3.2 Local DHT

The local DHT implementation is based on the Kademlia algorithm. Multiple local DHTs are created by partitioning nodes within the global DHT based on the files they store. Each local DHT is responsible for storing and querying the location information of data chunks related to a specific file, enabling quick localization and retrieval. This approach reduces the file retrieval time overhead, lowers network communication costs, and enhances the concurrency performance of the query system.

3.2.1 Storing Data Chunk CID

Since nodes in the local DHT only store and query the location information of data chunks related to a specific file, rather than the entire network's data chunks, each leaf node reports the data chunks it stores to the root node in the global DHT when a file is segmented into leaf nodes. The root node stores the CIDs of the data chunks, enabling other nodes to access the local DHT. The algorithm for storing data chunk CIDs is shown in Algorithm 1.

Algorithm 1: Storing the CID of the data chunk

```
Input: fileCID
Output: leafCIDs, leafNodes
1: Initialize a list leafCIDs as empty
2: Initialize a list leafNodes as empty
3: Query and return leafCIDs with the leaf hash of fileCID from globalDHT
4: For each CID in leafCIDs:
5:         Query globalDHT for nodes storing the data chunk with hash
6:         Add returned nodes to leafNodes
7: For each node in LeafNodes:
8:         For each CID in LeafCIDs:
9:                 If node stores data chunk with CID:
10:                        node.Store(self,fileCID,CID)
11:        End For
12: End For
13: Return leafCIDs, leafNodes
Function Store(file_cid,leaf_cid)
      closestnodes = FindClosestnodes(file_cid,K)
      For closenode in closestnodes
            closenode.storeCID(file_cid,leaf_cid)
      End For
Function: FindClosestNodes(hash, count)
       Query globalDHT for nodes storing the hash
       Sort nodes in localDHT based on XOR distance from key
       Return the first count nodes from the sorted list
Function storeCID(self,key,value)
       self.data_objects[key]['leaflinks'].append(value)
```

3.2.2 Establishing the Local DHT

After the leaf node stores the CID of the data chunk, the following steps are implemented to establish the local DHT and exchange location information:

(1) Local DHT initialization: Leaf nodes must store the location information of data chunks related to a specific file. They query the CID of the file in the global DHT, which then initializes a local DHT by returning the location information of the leaf nodes associated with the file's data chunks.

(2) Exchange of location information: Leaf nodes communicate with one another using the Kademlia algorithm to exchange location information. This enables each leaf node to know the positions of other relevant nodes and directly contact them to store or retrieve data.

(3) Data chunk indexing: Leaf nodes use the hash values of data chunks as keys, associating the keys with their own location information, and storing them in the local DHT.

(4) Consistency check: In the local DHT, nodes periodically update their routing tables and data chunk information to respond to changes in the network and maintain data reliability. Simultaneously, it is crucial to ensure consistency between the local DHT and the global DHT. To achieve this, nodes verify if the information in the local DHT is still present in the global DHT and update or delete it as necessary. The completed local DHT is shown in Fig. 5. This approach effectively organizes leaf nodes, enabling them to swiftly respond to queries related to specific files. This structure not only improves data retrieval efficiency but also enhances network scalability and robustness. The process for establishing and joining the local DHT is shown in Algorithm 2 and Algorithm 3, respectively.

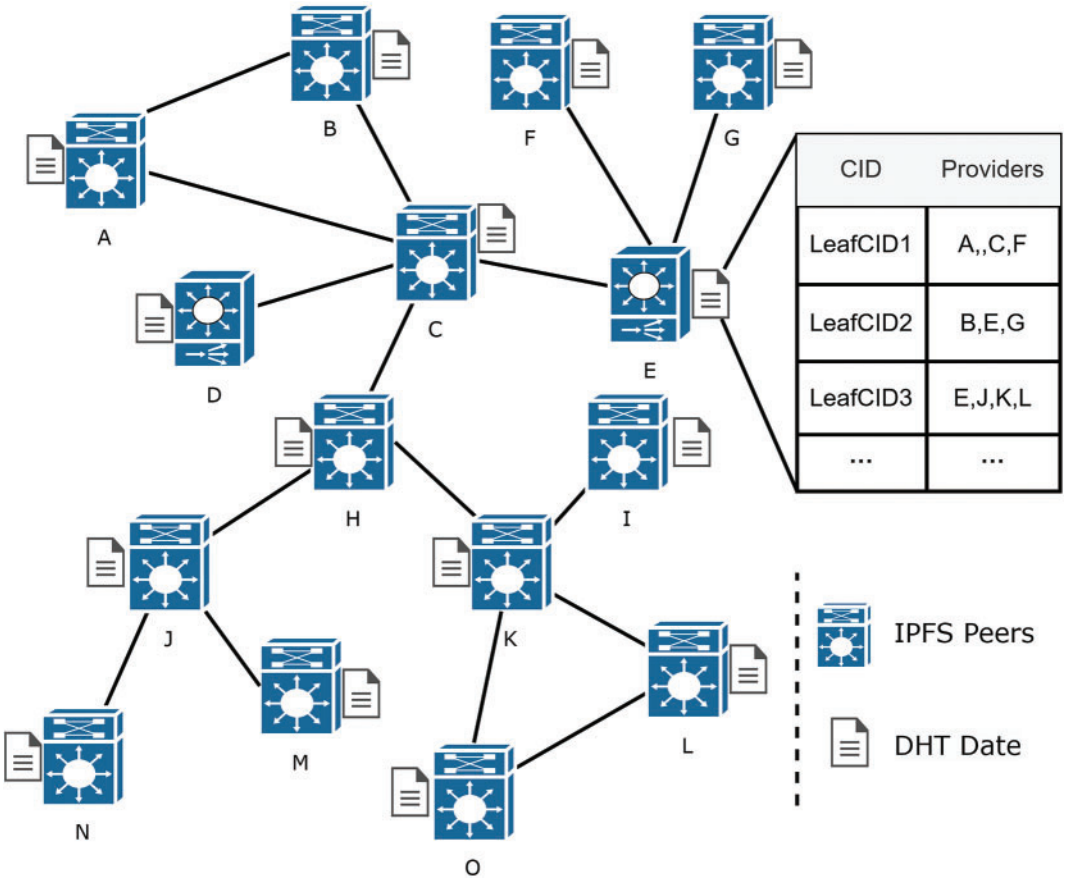


Figure 5: Local DHT establishment

Algorithm 2: Initializing the local DHT

```
Input:  leafCIDs, leafNodes
Output:  localDHT
1:  Initialize localDHT  as an empty hash tabl
2:  For each node in leafNodes:
3:          For each CID in leafCIDs:
4:                  If node stores data chunk with CID:
5:                          UpdateLocalDHT(node,CID)
6:          End For
7:  End For
8:  Return localDHT
Function: UpdateLocalDHT(node, hash)
        localDHT[hash] = node.networkInfo
                closenode.storeCID(file_cid,leaf_cid)
```

Algorithm 3: Joining the local DHT

```
Input: leafNodes, leafCIDs
Output: localDHT
1: For each node in leafNodes:
2:          For each CID in leafCIDs:
3:                  If node stores data chunk with CID:
4:                          JoinLocalDHT(node, CID)
5:          End For
6:  End For
7:  Return localDHT
Function:  JoinLocalDHT(node, hash)
        closestNodes  = FindClosestNodes(hash, K)
        For each closeNode  in closestNodes:
                Send join request to closeNode
                Receive and merge routing information from closeNode
        End For
        UpdateLocalDHT(node, hash)
```

3.2.3 Retrieving the Data Chunks

When retrieving a file, once the provider node for the file is found in the global DHT, the Merkle DAG structure of the file and the CIDs of its data chunks can be obtained. Subsequently, based on the CIDs of data chunks, the entry node is determined, and the location information of the data chunks is queried in the local DHT. Entry nodes refer to certain nodes within the local DHT, they serve as starting points for querying the data chunks. Once the entry node of the local DHT is identified, based on the CIDs of the data chunks, the nodes that store the data chunks are found to obtain the data, and then the data is reorganized into a complete file. The data chunk is retrieved as shown in Algorithm 4.

Algorithm 4: Retrieving the data chunk

```
Input:  leafCIDs
Output:  DataChunks
```

(Continued)

Algorithm 4 (continued)

```
Initialize DataChunks as an empty map
For each leafCID in leafCIDs:
    leafNode = FindClosestNodes(leafCID, 1)
    If leafNode.networkInfo is In LocalDHT:
        localDHT = leafNode.getLocalDHT()
        For each localCID in localDHT:
            localNode = localDHT.find(localCID)
            dataChunk = localNode.getDataChunk()
            DataChunks[localNode] = dataChunk
            If DataChunks can reorganize the entire file:
                Return DataChunks
    Else:
        dataChunk = leafNode.getDataChunk(leafCID)
        DataChunks[leafCID] = dataChunk
Return DataChunks
```

3.3 Summary

The relationship between the global DHT and the local DHT is complementary and coordinated. Each node has its position and role in the global DHT, while it may also play specific roles in one or more local DHTs. The key points of their relationship are as follows:

(1) Global DHT: The global DHT forms the foundation of the entire IPFS network, responsible for maintaining information on all nodes and the locations of data chunks in the network. It functions as a large-scale distributed hash table, in which every node participates.

(2) Local DHT: The local DHT is created for specific files or sets of files, consisting of nodes that store related data chunks. The purpose of the local DHT is to optimize the retrieval and storage processes for data associated with these specific files.

(3) Dual roles of nodes: A node can exist simultaneously in the global DHT and one or more local DHTs. In the global DHT, nodes are responsible for maintaining the overall network health and data accessibility. In the local DHT, nodes focus on managing data chunks related to specific files.

(4) Independence and coordination: The local DHT is somewhat independent of the global DHT, but its nodes still rely on the global DHT to discover and join the network. The establishment and maintenance of the local DHT also require synchronization with the global DHT to ensure the consistency and efficiency of the entire network.

In summary, local DHTs form sub-networks for specific tasks, based on the global DHT. They coordinate by sharing nodes and partial routing information. This design enhances IPFS network performance and response speed while maintaining decentralization and scalability.

4 Copyright Protection Scheme Design

4.1 Overall Design

Fig. 6 shows the architecture diagram of the proposed scheme. There are two entities in the scheme, that is, the image owner and the verifier. The numbers identified in the figure represent the steps of the scheme, which will be described in Section 4.2.

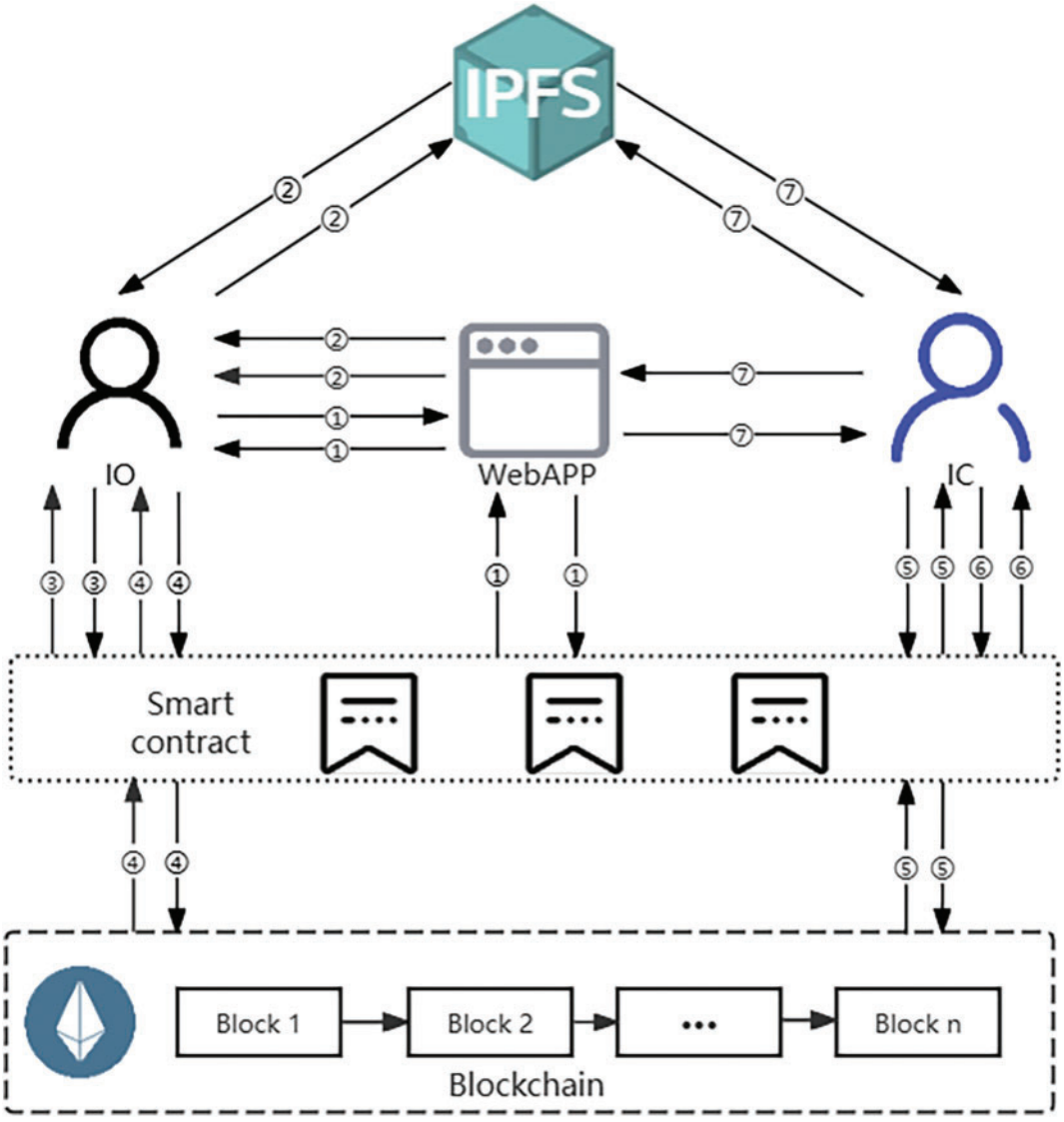


Figure 6: Overall architecture

The components of this scheme can be primarily divided into three parts. The first part is the blockchain network, which includes the smart contract interface and blockchain storage. The blockchain stores relevant image information. As a decentralized database, the blockchain securely stores data without requiring trusted third parties. It provides a transparent and tamper-proof storage platform that ensures the authenticity and reliability of copyright information. Smart contracts deployed on the blockchain include functions such as user registration, information uploading, and signing, which are agreed upon and executed by all nodes. Corresponding smart contracts are designed for different functions, with their interfaces shown in Table 1.

The second part, the WebApp, processes the specific operations of the entities and is mainly responsible for embedding and extracting blind watermarks. Blind watermarks are used to add identity tags to images to prevent unauthorized re-uploading and tampering. When embedding blind

watermarks, the entity can adjust the watermark strength parameter as needed. A higher strength parameter enhances watermark resistance but may affect the original image quality, while a lower strength parameter has the opposite effect. Additionally, the WebApp can invoke smart contracts to coordinate the work between IPFS and the blockchain.

Table 1: Smart contract interface

Contract name	Interface definition	Interface description
Register	userRegister	User registration
StoreInfo	setInfo	Store information
	getInfo	Query information
PersonalSign	getSign	Sign message
	verifySign	Verify signature

The third part, IPFS, stores image files with embedded blind watermarks and generates a unique CID. When it's necessary to retrieve image files and verify watermark information, the file is retrieved from the IPFS network using the CID.

Since IPFS uses DHT to manage file storage location, the dual-layer DHT indexing structure proposed in the paper further optimizes the file retrieval process. Therefore, the decentralized storage and efficient retrieval provided by IPFS, together with the blockchain, create an efficient and secure copyright protection mechanism. IPFS manages the file storage and retrieval, ensuring their integrity and availability. Blockchain is responsible for recording and verifying copyright information, ensuring its immutability and openness and transparency. This synergistic effect not only improves the efficiency of file retrieval and copyright management, but also enhances the scalability and stability of the copyright protection scheme.

4.2 Scheme Workflow

The symbols used in the text are as shown in Table 2.

Table 2: Related symbol specification

Symbol	Definition
IO	Image owner
IC	Verifier
P_k	Public key (Account address)
S_k	Private key
$CID( )$	Content-addressed hash value
I	Image file
WI	Image file with embedded watermark
Sig	Digital signature
wm	Watermark information
TXH	Transaction hash

As shown in Fig. 6, Steps ① to ⑦ outline the specific process of the scheme as follows:

(1) Image Owner

① The Image Owner (IO) initiates the registration request by entering the password and other basic information through the WebApp. WebApp generates the public-private key pair (P_k) and (S_k) of IO according to the entered password, and invokes Registr.userRegister to complete registration.

② IO selects the image file I and sets the watermark strength parameter in the WebApp. Then, IO embeds the public key P_k as watermark information wm into I, generating the watermarked image file WI. IO uploads WI to the IPFS network, obtaining the content hash value $CID(WI)$.

③ IO invokes PersonalSign.getSign, which uses S_k to sign $CID(WI)$, generating the signature Sig.

④ IO invokes StoreInfo.setInfo to store P_k, $CID(WI)$, Sig, and other ownership information on the blockchain. The smart contract confirms the transaction with IO and sends the transaction hash TXH to IO. The blockchain network broadcasts the transaction request. After achieving consensus among blockchain nodes, the transaction is packaged, and the information is permanently stored on the blockchain.

(2) Verifier

⑤ The Verifier (IC) invokes StoreInfo.getInfo to query the transaction information associated with TXH. The smart contract retrieves transaction information from the blockchain, including P_k, $CID(WI)$, and Sig.

⑥ Based on the obtained information, IC invokes PersonalSign.verifySign to verify the signature. If the signature verification is successful, it proves that the signature was signed by the holder of the private key S_k, confirming P_k as the genuine uploader of the image. Otherwise, the verification fails, and the signature does not belong to the IO.

⑦ IC retrieves the corresponding image file WI' from the IPFS network using $CID(WI)$, then extracts the watermark information wm' from WI' using the WebApp. IC checks whether the extracted watermark wm' matches the public key P_k. If $wm' = P_k$, it confirms P_k as the rightful owner of the file. Otherwise, the file is considered unauthorized use.

The entire process of the scheme can be divided into five stages:

(1) User registration: To generate signatures and invoke smart contracts, entities need to possess key pairs associated with the account. IO obtains its public and private keys through the WebApp.

(2) Image storage: IO embeds the public key into the image as watermark information using blind watermark embedding algorithms via WebApp, adding an anti-counterfeiting identification to the image, which results in a watermarked image file. The file is then stored in IPFS, and its CID is obtained.

(3) Signing: IO generates the file CID's signature using its private key through signature algorithms.

(4) Information storage: After signing, relevant ownership information is stored in the blockchain, ensuring security and credibility through the blockchain's immutability and decentralization. Other entities can verify the signature based on the information on the blockchain.

(5) Ownership verification: To verify image ownership, verifiers first retrieve relevant transaction information from the blockchain and use this information to confirm the signature's authenticity. Once the signature is verified, verifiers proceed to verify the watermark information of the file. IC uses blind

watermark extraction algorithms via WebApp to extract watermark information and compare it with the public key, determining whether the file is unauthorized.

4.3 Algorithm Design

Given the advantages of the ECDSA (Elliptic Curve Digital Signature Algorithm) [23], including small key size, short signature generation time, and fast computation speed, this paper adopts ECDSA for digital signature generation and verification. At the same time, this paper implements the embedding and extraction of watermark information based on the Discrete Wavelet Transform-Discrete Cosine Transform-Singular Value Decomposition (DWT-DCT-SVD) blind watermarking algorithm. Compared with traditional DWT and DCT algorithms, this approach exhibits better robustness, effectiveness, and imperceptibility [24].

Singular Value Decomposition (SVD) [25] is applied in orthogonal matrices, serving as a linear algebra tool. Let matrix $A \in R^{m*n}$, where R represents the real number field, and $m * n$ denotes the matrix size. The SVD of A is defined as follows:

$$A = USV^T \tag{2}$$

where $U = [u_1, u_2, u_3, \ldots, u_m] \in R^{m*m}$, $V = [v_1, v_2, v_3, \ldots, v_m] \in R^{n*n}$, and T represents the transpose. U and V are orthogonal matrices. S is a diagonal matrix, $S = \text{diag}\,(\sigma_1, \sigma_2, \sigma_3, \ldots \sigma_m) \in R^{m*n}$, where the singular values ($\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \sigma_r \geq \sigma_{r+1} = \ldots = \sigma_m = 0$) are the diagonal elements, with r representing the rank of the matrix.

Let the size of the image file I be 512×512 pixels. The watermark information, denoted as *wm*, is the account address and is defined as a string type.

4.3.1 Watermark Embedding

Step 1: Convert the watermark information *wm* into binary form, obtaining a binary array *wi*.

Step 2: Extract the Red, Green, and Blue (RGB) components of image$I$and convert them into the Luminance and Chrominance (YUV) format. Perform a level-one DWT on the Y component to obtain four sub-bands: LL, HL, LH, and HH. Divide the low-frequency sub-band LL into 8×8 blocks, with each block embedding 1 bit of data. Thus, the maximum number of watermark bits that can be embedded in the low-frequency sub-band LL is calculated as $512/(2 \times 8) \times 512/(2 \times 8) = 1024$.

Step 3: Apply DCT to each block, followed by SVD on the output matrix of DCT, $Block_i = U_i S_i V_i^T$, and arrange the singular values σ_i in descending order.

Step 4: Modify the first σ_1 in each block according to the following rules (let $Z = \sigma_1 \text{mod} \beta$):

① $w_i = 0$, then:

$$\begin{cases} \sigma_1' = \sigma_1 - Z + 5\beta/4, \text{ if } Z \geq 3\beta/4 \\ \sigma_1' = \sigma_1 - Z + \beta/4, \text{ else} \end{cases} \tag{3}$$

② $w_i = 1$, then:

$$\begin{cases} \sigma_1' = \sigma_1 - Z + 3\beta/4, \text{ if } Z \geq \beta/4 \\ \sigma_1' = \sigma_1 - Z - \beta/4, \text{ else} \end{cases} \tag{4}$$

where β is the watermark embedding strength factor, and the modified block is $Block_i' = U_i S_i' V_i^T$.

Step 5: Repeat the operations of Steps 3 and 4 until all watermark information is embedded. Then perform inverse DCT to obtain the low-frequency sub-band with embedded watermark, followed by

inverse DWT to obtain the Y component with embedded watermark. Finally, convert the YUV format back to the RGB format and reconstruct the RGB components to obtain the watermarked image *WI*. The watermark embedding process is illustrated in Fig. 7.

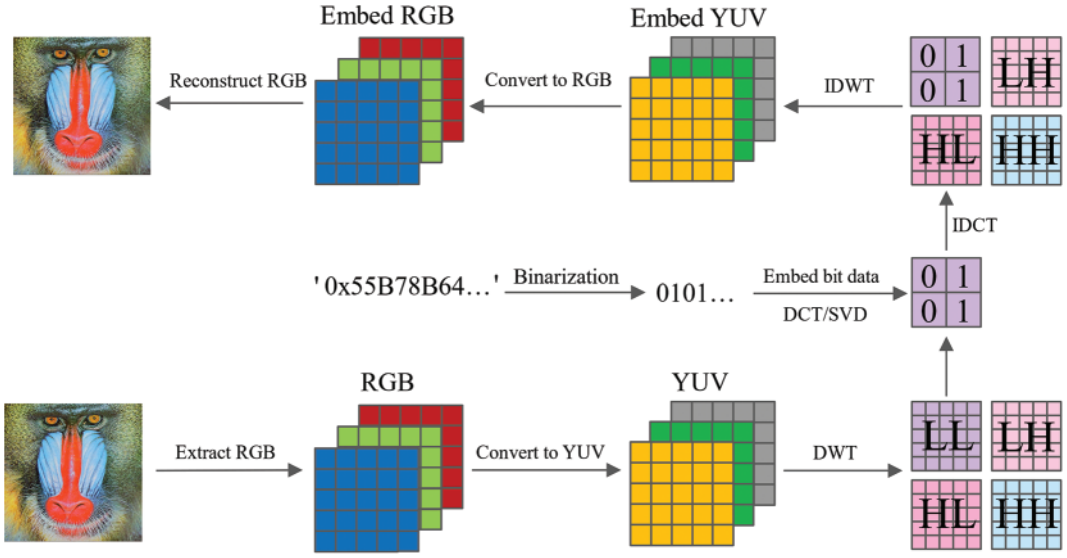


Figure 7: Watermark embedding

4.3.2 Watermark Extraction

Step 1: Extract the RGB components of the image *WI*, convert them from RGB to YUV format, and perform a level-one DWT on the Y component. Then divide the low-frequency sub-band LL into 8×8 blocks.

Step 2: Apply DCT to each block and perform SVD on the output matrix, $Block_i' = U_i S_i' V_i^T$, and arrange the singular values σ_i' in descending order.

Step 3: Modify w_i according to the following rule (let $Z = \sigma_1' \bmod \beta$):

$$\begin{cases} w_i = 0, \text{ if } Z \leq \beta/2 \\ w_i = 1, \text{ else} \end{cases} \tag{5}$$

Step 4: Repeat the operations of Steps 2 and 3 until all embedded watermark information is extracted from the low-frequency sub-band.

Step 5: Convert the extracted w_i back into a string format to obtain the original watermark information *wm*. The watermark extraction process is illustrated in Fig. 8.

5 Experiment and Analysis

The experiments were conducted on a local area network, consisting of a private IPFS network cluster consisting of 20 nodes, and a private blockchain built locally using Geth. The specific experimental environment is shown in Table 3.

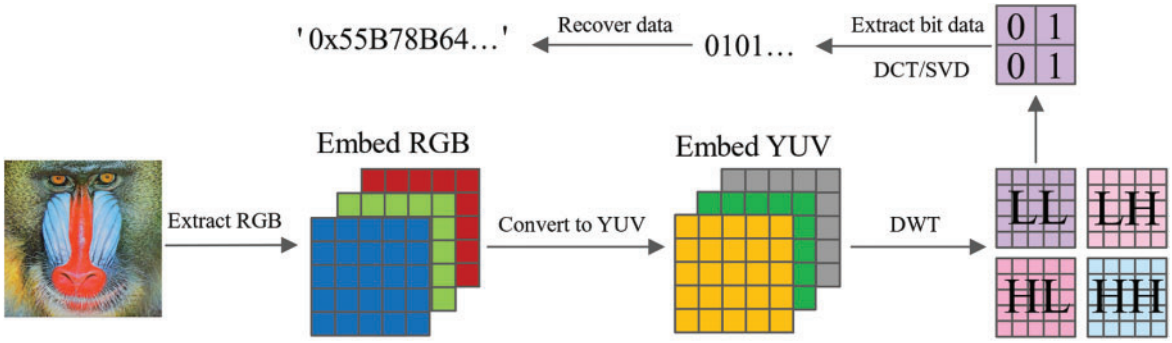


Figure 8: Watermark extraction

Table 3: Experimental environment

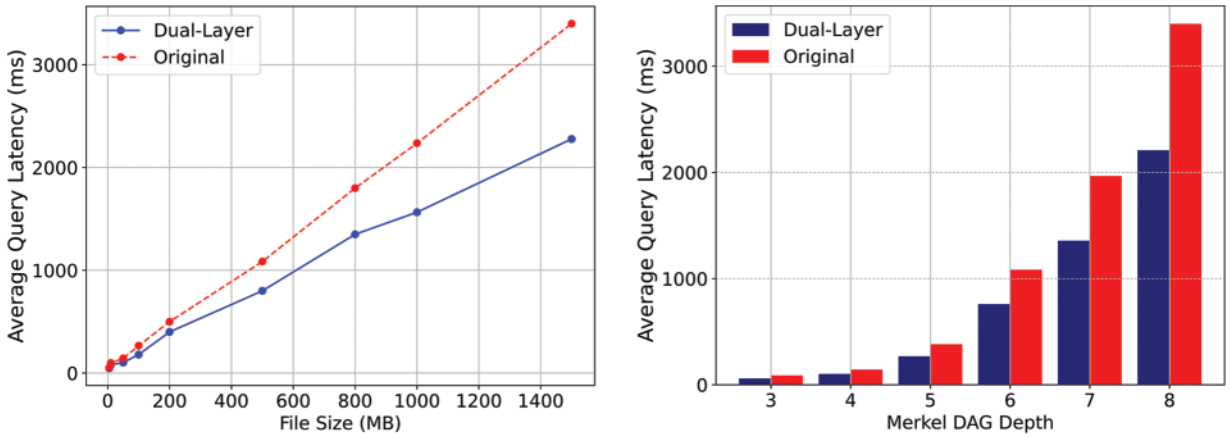
Category	Configuration/Version
CPU (Central Processing Unit)	13th Intel(R) Core(TM) i7-13620H 2.40 GHz
Internal memory	16 GB
Operating system	Ubuntu, v22.04.1
IPFS	Kubo, v0.22.0
Blockchain	Geth, v1.9.10

5.1 Dual-Layer DHT Indexing Structure Analysis

Setting the data chunk size to the default 256 KB, as shown in Fig. 9, the use of the dual-layer DHT indexing structure method results in overall reduced DHT query time compared to IPFS's native DHT. In Fig. 9a, the reduction in DHT latency is minimal for small files. However, as the file size increases, the optimization effect of the dual-layer DHT indexing structure becomes more pronounced, reducing query latency by approximately 33% compared to the native DHT. The reduction effect grows as file size increases. This occurs because as file size increases, the number of data chunks also grows, leading to more intermediate nodes in the Merkle DAG and greater depth. The dual-layer DHT indexing structure reduces queries to intermediate nodes in the Merkle DAG, thereby significantly lowering the overall DHT query time. The impact of Merkle DAG depth on query latency is shown in Fig. 9b, where the optimization effect of the dual-layer DHT indexing structure becomes significant at a depth of 5 and intensifies as the depth increases. To confirm the impact of Merkle DAG depth on query latency, in Fig. 10, the Merkle DAG depth was varied by adjusting the data chunk size. Files of 50, 100, 500, and 1000 MB were constructed to test the changes in query latency, as shown in Fig. 11.

Compared to the existing methods, the proposed dual-layer DHT indexing structure not only improves the efficiency but also significantly enhances the scalability and reliability. Zhu et al. [19] employed distributed B+ Tree and HashMap for keyword searches, but the complexity of their index merging algorithm and network traffic adversely affected system performance. Cao et al. [20] proposed a double-layer inverted indexing structure that improved search efficiency but imposed significant update overheads, thereby limiting system scalability. Khudhur et al. [21] used DHT to store mappings between keywords and CIDs, and leveraged result caching to speed up repeated

queries. However, their approach offered limited improvements for new queries and faced scalability challenges. Yao et al. [22] employed Fat Merkle Trees and DHT caching mechanisms to reduce query latency, but their complex structure increased operational burdens, impacting system reliability. In contrast, the dual-layer DHT indexing structure simplifies the query process and reduces network overhead, providing a more effective solution to the problem of low IPFS file retrieval efficiency.



(a) Query latency for different file sizes

(b) Query latency at different Merkel DAG depths

Figure 9: Query latency

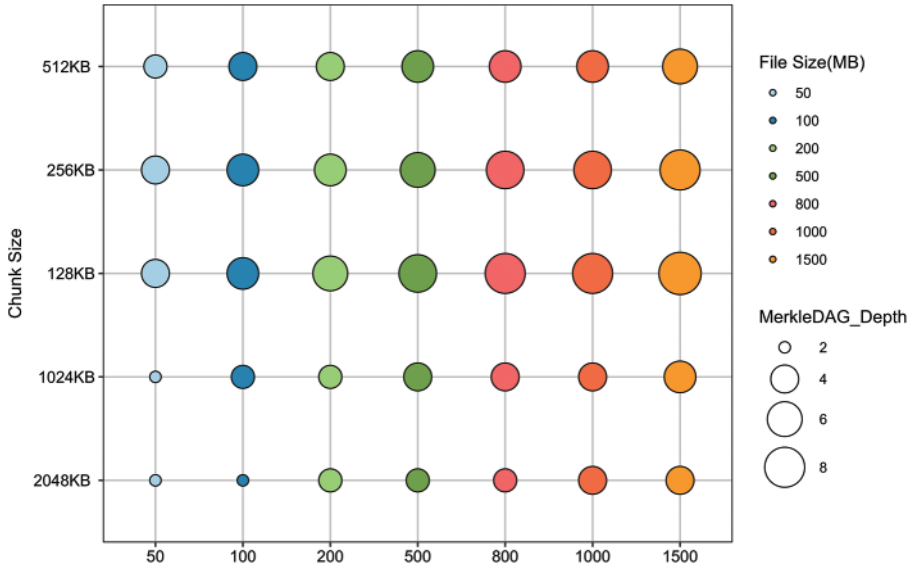


Figure 10: Merkle DAG depth changes

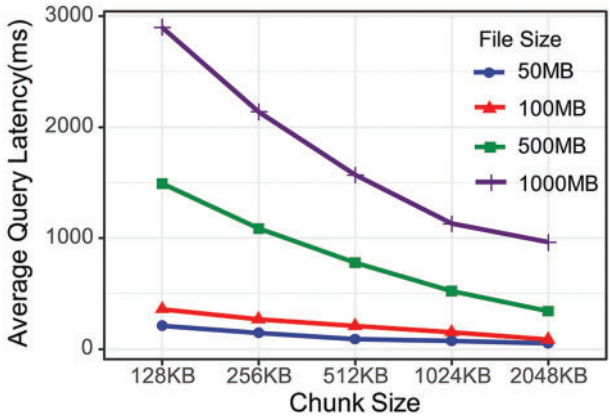


Figure 11: Query latency for different chunk sizes

5.2 Copyright Protection Scheme Analysis

5.2.1 Ownership Analysis of Files

(1) Trusted source information for files

Unlike traditional centralized technologies, this scheme leverages blockchain and smart contracts to ensure the security of source information for image files. By recording information such as CID, account addresses, and digital signatures on the blockchain, automatically managed by smart contracts, the risk of third-party data tampering is eliminated. This approach provides trusted and verifiable transaction records of file ownership. Once the information is recorded, it is permanently associated with the file, meaning that account addresses and signature information can be reliably traced and verified. Smart contracts provide automated interfaces that simplify file ownership verification, ensuring reliable information circulation and transparency throughout the verification process. This creates a robust trust foundation for file ownership verification. Additionally, this framework is flexible and scalable, capable of adapting to various user needs.

(2) Identity verification and copyright tracing

The scheme employs blind watermarking to embed the image owner's account address into the image as watermark information. This effectively deters unauthorized use and copyright infringement, while enabling the owner to track and verify their work. Should unauthorized distribution occur, the owner can extract the watermark to reveal the account address, trace the blockchain record, and thereby establish the image's true ownership. The privacy of the image owner is strictly protected, as the account address is only disclosed during ownership verification.

By combining digital signature technology, the owner's account address is closely linked to the image's CID. The uniqueness of the digital signature ensures that only the image owner can generate a valid signature. During signature verification, if both the CID and the account address are correct, the verification succeeds, and the account address is confirmed as the true uploader. Any modification results in verification failure. By comparing the extracted watermark information from the IPFS image with the account address, ownership can be determined, thereby establishing the true owner of the image. This mechanism not only verifies the true source of the image but also significantly enhances strengthens copyright protection.

5.2.2 *Performance Analysis*

This experiment section aims to evaluate the impact of blind watermark on the image upload time to IPFS, as well as the performance of the proposed scheme across different processing stages. Five test images of varying sizes were selected, and a blind watermark was embedded in each. Both the original images and watermarked images were uploaded to IPFS, with the upload times recorded as shown in Fig. 12. Finally, the selected images were applied to the entire copyright protection scheme, and the average runtime of each key stage was measured. The data in Fig. 12 shows that the time required to upload images to IPFS differs by less than 50 ms before and after embedding a blind watermark. This indicates that the embedding blind watermark has a negligible impact on IPFS upload time, allowing users to do so without concern for significantly increasing the upload duration. As shown in Table 4, the average runtime of each stage of the scheme is within milliseconds, and the time spent is acceptable and not disruptive to daily applications.

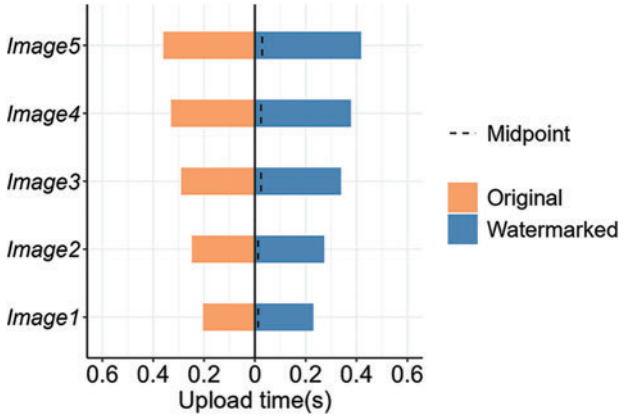


Figure 12: Upload time

Table 4: Average running time for different phases

Phase	Average run time (ms)
User registration	84
Storage of image	352
Signing	68
Storing information	836
Ownership verification	781

5.2.3 *Smart Contract Testing*

Detailed gas consumption tests were conducted on the contract to evaluate the execution costs. Table 5 lists the costs associated with executing the contract. The contract is deployed once during the blockchain network's initialization and does not require redeployment during normal operations, resulting in no additional gas consumption. The setInfo and userRegister functions within the contract generate transactions and incur gas costs, whereas other functions, which do not alter the contract

state, do not incur gas costs. The experiments indicate that the cost of the scheme is relatively low, providing substantial advantages for applications.

Table 5: Contract cost consumption

Operation	Gas cost (wei)	Ether cost (ETH)
Deploy register	138684	0.000138684
Deploy StoreInfo	347716	0.000347716
Deploy PersonalSign	288710	0.000288710
userRegister	43455	0.000043455
setInfo	53187	0.000053187

5.2.4 Scheme Comparison

Table 6 presents a comprehensive performance comparison between the proposed research scheme and existing schemes that employ IPFS for the protection of image file copyright. While schemes [10] and [11] were designed to create image copyright management systems that would ensure the security and transparency of copyright information, they lacked the capacity to be scaled up. Scheme [12] detected infringement through perceptual hash technology but did not fully address user ownership issues. Scheme [13] not only completed identity verification but also employed dual encryption storage technology to mitigate the risk of copyright information theft. However, the verification process was relatively complex. Scheme [14] enhanced the security and transparency of copyright information, and provided robust copyright protection for multiple creations, however, it lacked scalability. Schemes [15] and [16] authenticated images by designing smart contracts and combining them with zero watermark algorithms while also ensuring the scalability of the image copyright protection system through IPFS. However, limitations were encountered with respect to file retrieval. As shown in Table 6, the proposed scheme demonstrates superior performance in terms of scalability, privacy protection, and identity verification in comparison to existing schemes.

Table 6: Comparison of existing copyright protection schemes

Scheme	Scalability	Privacy	Authentication	Infringement detection	Security
[10]	×	✓	✓	None	×
[11]	×	✓	×	Perceptual hashing	×
[12]	×	✓	×	Perceptual hashing	✓
[13]	✓	✓	✓	Perceptual hashing	✓
[14]	×	✓	✓	Watermark	✓
[15]	✓	✓	✓	Watermark	✓
[16]	✓	✓	✓	Watermark	✓
Proposed	✓	✓	✓	Watermark	✓

Existing schemes generally employ IPFS to store images or copyright information. However, there is still scope for improvement in terms of the efficiency of retrieving IPFS files. The proposed research scheme addresses this issue by optimizing the efficiency of IPFS file retrieval, thereby resolving

potential performance bottlenecks encountered by existing schemes in practical applications. This improvement ensures rapid access to image files and copyright information.

6 Conclusion

This paper addresses the issue of copyright protection for image files on IPFS. The proposed approach organically combines blockchain, digital signature, and blind watermarking technology to successfully ensure copyright protection of images, effectively prevent unauthorized copying and tampering, and achieve infringement prevention. The smart contract method is designed and applied with the objective of ensuring the secure storage of the ownership information and the identity verification of the uploader. This further enhances the security and trustworthiness of the information storage and verification process. Additionally, to address the efficiency issues in IPFS querying, a dual-layer DHT indexing structure is proposed. Through experimental analysis, it has been demonstrated that the efficiency and performance of file queries have been enhanced significantly under the dual-layer DHT indexing structure, the latency of file retrieval has been greatly reduced, and the success rate of query has been increased.

Acknowledgement: We would like to acknowledge the editors and anonymous reviewers.

Funding Statement: This work was supported by the Doctoral Research Foundation of Chongqing Normal University (Nos. 21XLB030, 21XLB029) and the Key Program of Chongqing Education Science Planning Project (No. K22YE205098).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Xin Cong, Lanjin Feng; methodology: Xin Cong; data collection: Lanjin Feng; analysis and interpretation of results: Lingling Zi; writing—review & editing: Xin Cong, Lingling Zi, Lanjin Feng. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The authors confirm that the data and materials supporting the findings of this study are not applicable as there were no specific datasets or materials used.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. Chen, H. Li, K. Li, and J. Zhang, "An improved P2P file system scheme based on IPFS and Blockchain," in *2017 IEEE Int. Conf. Big Data*, Boston, MA, USA, Jan. 2017, pp. 2652–2657.

[2] P. Kang, W. Yang, and J. Zheng, "Blockchain private file storage-sharing method based on IPFS," *Sensors*, vol. 22, no. 14, pp. 5100–5112, Jul. 2022. doi: 10.3390/s22145100.

[3] Y. Shang, "Efficient and secure algorithm: The application and improvement of ECDSA," in *2022 Int. Conf. Big Data, Inf. Comput. Netw. (BDICN)*, Sanya, China, Jan. 2022, pp. 182–188.

[4] J. Wang, D. Wu, L. Li, J. Zhao, H. Wu and Y. Tang, "Robust periodic blind watermarking based on sub-block mapping and block encryption," *Expert. Syst. Appl.*, vol. 224, Aug. 2023, Art. no. 119981. doi: 10.1016/j.eswa.2023.119981.

[5] S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, "Mapping the interplanetary filesystem," in *2020 IFIP Netw. Conf.*, Paris, France, Jun. 2020, pp. 289–297.

[6] L. Chen, X. Zhang, and Z. Sun, "Scalable blockchain storage model based on DHT and IPFS, KSII Trans," *Internet Inf. Syst.*, vol. 16, no. 7, pp. 2286–2304, Jul. 2022. doi: 10.3837/tiis.2022.07.009.
[7] C. Chrysoulas *et al.*, "GLASS: Towards secure and decentralized eGovernance services using IPFS," in *Eur. Symp. Res. Comput. Secur.*, Darmstadt, Germany, Oct. 2021, pp. 40–57.
[8] S. Alghamdi, S. Naz, A. Saeed, E. A. Solami, M. Kamran and M. S. Alkatheiri, "A novel database watermarking technique using blockchain as trusted third party," *Comput. Mater. Contin.*, vol. 70, no. 1, pp. 1585–1601, Sep. 2022. doi: 10.32604/cmc.2022.019936.
[9] X. Chen, A. Yang, J. Weng, Y. Tong, C. Huang and T. Li, "A Blockchain-based copyright protection scheme with proactive defense," *IEEE Trans. Serv. Comput.*, vol. 16, no. 4, pp. 2316–2329, Jun. 2023. doi: 10.1109/TSC.2023.3246476.
[10] J. Ning, L. Feng, G. Yuan, and X. Yang, "Research on image trading system based on Blockchain and IPFS," in *Int. Conf. Netw. Commun. Inf. Secur. (ICNCIS)*, Beijing, China, Apr. 2022, pp. 51–58.
[11] A. Muwafaq and S. N. Alsaad, "Design scheme for copyright management system using Blockchain and IPFS," *Int. J. Comput. Digit. Syst.*, vol. 10, no. 1, pp. 613–618, May 2021. doi: 10.12785/ijcds/100159.
[12] R. Kumar, R. Tripathi, N. Marchang, G. Srivastava, T. R. Gadekallu and N. N. Xiong, "A secured distributed detection system based on IPFS and blockchain for industrial image and video data security," *J. Parallel Distr. Comput.*, vol. 152, no. 2, pp. 128–143, Jun. 2021. doi: 10.1016/j.jpdc.2021.02.022.
[13] Q. Zhang, G. Wu, R. Yang, and J. Chen, "Digital image copyright protection method based on blockchain and zero trust mechanism," *Multimed. Tools Appl.*, vol. 17, no. 2, pp. 1–36, Feb. 2024. doi: 10.1007/s11042-024-18514-3.
[14] Z. Meng, T. Morizumi, S. Miyata, and H. Kinoshita, "Design scheme of copyright management system based on digital watermarking and blockchain," in *2018 IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Tokyo, Japan, Jul. 2018, pp. 359–364.
[15] B. Wang, J. W. Shi, W. Wang, and P. Zhao, "Image copyright protection based on blockchain and zero-watermark," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2188–2199, May 2022. doi: 10.1109/TNSE.2022.3157867.
[16] T. Chen *et al.*, "A image copyright protection method using zero-watermark by blockchain and IPFS," *J. Inf. Hiding Priv. Prot.*, vol. 3, no. 3, pp. 131–142, Dec. 2021. doi: 10.32604/jihpp.2021.026606.
[17] S. G. Liu, W. Q. Chen, and J. L. Liu, "An efficient double parameter elliptic curve digital signature algorithm for blockchain," *IEEE Access*, vol. 9, pp. 77058–77066, May 2021. doi: 10.1109/ACCESS.2021.3082704.
[18] D. Trautwein *et al.*, "Design and evaluation of IPFS: A storage layer for the decentralized web," in *Proc. ACM SIGCOMM*, Amsterdam, Netherlands, Aug. 2022, pp. 739–752.
[19] L. Zhu, C. Xiao, and X. Gong, "Keyword search in decentralized storage systems," *Electronics*, vol. 9, no. 12, pp. 2041–2058, Dec. 2020. doi: 10.3390/electronics9122041.
[20] L. Cao and Y. Li, "IPFS keyword search based on double-layer index," in *Int. Conf. Electron. Inf. Eng. Comput. Commun. (EIECC)*, Nanchang, China, May 2022, pp. 45–50.
[21] N. Khudhur and S. Fujita, "Siva-The IPFS search engine," in *2019 Seventh Int. Symp. Comput. Netw. (CANDAR)*, Nagasaki, Japan, Nov. 2019, pp. 150–156.
[22] Z. Yao, B. Ding, Q. Bai, and Y. Xu, "Minerva: Decentralized collaborative query processing over interplanetary file system," *IEEE Trans. Big Data*, pp. 1–15, Jul. 2024. doi: 10.1109/TBDATA.2024.3423729.
[23] S. Ullah, J. Zheng, N. Din, M. T. Hussain, F. Ullah and M. Yousaf, "Elliptic curve cryptography; Applications, challenges, recent advances, and future trends: A comprehensive survey," *Comput. Sci. Rev.*, vol. 47, pp. 100530–100544, Feb. 2023. doi: 10.1016/j.cosrev.2022.100530.
[24] S. Bagheri Baba Ahmadi, G. Zhang, S. Wei, and L. Boukela, "An intelligent and blind image watermarking scheme based on hybrid SVD transforms using human visual system characteristics," *Vis. Comput.*, vol. 37, no. 2, pp. 385–409, Feb. 2021. doi: 10.1007/s00371-020-01808-6.
[25] H. T. Hu, L. Y. Hsu, and H. H. Chou, "An improved SVD-based blind color image watermarking algorithm with mixed modulation incorporated," *Inf. Sci.*, vol. 519, pp. 161–182, May 2020. doi: 10.1016/j.ins.2020.01.019.