**IPFS Based File Storage Access Control and Authentication Model for Secure Data Transfer using Blockchain Technique**

**ABSTRACT:**

Large files cannot be efficiently stored on blockchains. On one hand side, the blockchain becomes bloated with data that has to be propagated within the blockchain network. On the other hand, since the blockchain is replicated on many nodes, a lot of storage space is required without serving an immediate purpose, especially if the node operator does not need to view every file that is stored on the blockchain. It furthermore leads to an increase in the price of operating blockchain nodes because more data needs to be processed, transferred and stored. IPFS is a file sharing system that can be leveraged to more efficiently store and share large files. It relies on cryptographic hashes that can easily be stored on a blockchain. Nonetheless, IPFS does not permit users to share files with selected parties. This is necessary, if sensitive or personal data needs to be shared. Therefore, this paper presents a modified version of the Interplanetary Filesystem (IPFS) that leverages Ethereum smart contracts to provide access-controlled file sharing. The smart contract is used to maintain the access control list, while the modified IPFS software enforces it. For this, it interacts with the smart contract whenever a file is uploaded, downloaded or transferred. Using an experimental setup, the impact of the access controlled IPFS is analyzed and discussed.

**Keywords:** 1. Blockchain efficiency,2. IPFS file sharing, 3. Cryptographic hashes,4. Access-controlled sharing, ,6. Data security.

**STATEMENT ABOUT THE PROBLEM**

The integration of blockchain with file storage, particularly in large files, presents a critical challenge. Storing substantial data on blockchains results in network bloat and increased operational costs for mining nodes. Alternatively, using smart contracts for file storage incurs high transaction costs and strains network bandwidth. Additionally, public accessibility to files on platforms like IPFS poses privacy concerns, especially for sensitive data. To address these issues, this paper focuses on developing acl-IPFS, a modified InterPlanetary File System (IPFS) leveraging the Ethereum blockchain. The objective is to establish access control for IPFS files through an Ethereum smart contract, ensuring secure and permission-controlled file sharing.

**SCOPE:**

This project aims to address the inefficiencies and challenges associated with storing large files on traditional blockchains, specifically focusing on the InterPlanetary File System (IPFS). Leveraging the Ethereum blockchain, the project introduces an access-controlled IPFS system (acl-IPFS) through a smart contract. The scope includes the design and implementation of acl-IPFS, enabling users to register, access, and manage large files securely. The system allows dynamic modification of access control lists, granting and revoking permissions via Ethereum transactions. By establishing a secure and permissioned layer over IPFS, the project enhances the viability of blockchain applications dealing with sensitive or personal data in large files.

**OBJECTIVE OF THE PROJECT:**

This project aims to address the inefficiencies and limitations of utilizing blockchain for large file storage by leveraging the Ethereum blockchain to enhance the InterPlanetary File System (IPFS). The objective is to develop and implement an access-controlled IPFS solution, named acl-IPFS, using Ethereum smart contracts. By dynamically managing access control lists on the blockchain, acl-IPFS enables secure and permission-based file sharing. The project seeks to optimize file storage and retrieval, ensuring confidentiality for sensitive data, and establishing a robust linkage between IPFS nodes and Ethereum accounts for permission enforcement.

**EXISTING METHOD**

The existing system involves blockchain applications interacting with smart contracts for achieving consensus on transactions, data, or code execution. However, the system faces challenges when dealing with large data files, as storing them directly on the blockchain leads to inefficiency and bloating. Smart contracts can be used to store file parts, but this incurs high costs in terms of gas expenses and increased operational expenses for mining nodes. To address this, the paper proposes leveraging the InterPlanetary File System (IPFS) for efficient file sharing while keeping the blockchain size manageable. Additionally, the paper introduces acl-IPFS, a modified IPFS software integrated with an Ethereum smart contract for access-controlled IPFS.

**DISADVANTAGES**

**1. Scalability Concerns:**

As the number of files and users increases, the scalability of the IPFS network may become a challenge. The decentralized nature of IPFS relies on nodes to store and retrieve files, and the system may experience bottlenecks as the demand for storage and access control grows.

**2. Dependency on Ethereum Blockchain:**

The proposed model relies on the Ethereum blockchain for access control, introducing a dependency on the Ethereum network's performance and scalability. Fluctuations in Ethereum transaction costs and network congestion could impact the efficiency and cost-effectiveness of the file storage and access control system.

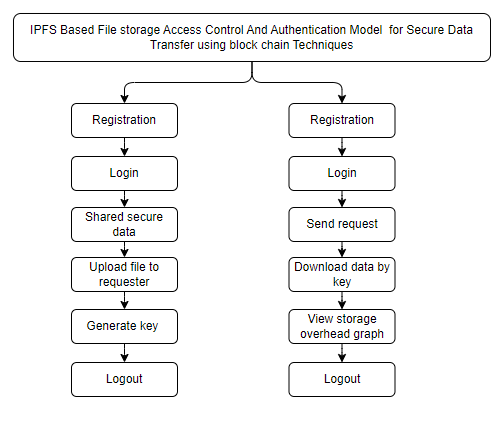
**3. Latency in File Retrieval:**

Access control enforcement through smart contracts on the Ethereum blockchain may introduce latency in retrieving files from IPFS. The need to interact with the blockchain for every access request can result in delays, especially during periods of high network activity or congestion.

**PROPOSED SYSTEM:**

This research introduces a novel system, acl-IPFS, designed to address the challenges of securely sharing large files on the Interplanetary File System (IPFS) while maintaining access control. Leveraging Ethereum blockchain technology, acl-IPFS employs a smart contract to store and dynamically manage an access control list. Users can register, grant, and revoke permissions for files, ensuring secure and controlled access. Each file request involves the acl-IPFS nodes providing a public key and signing the message with a linked Ethereum account, establishing a robust connection between nodes and the smart contract to enforce permissions. This innovative approach enhances file sharing security on IPFS within a decentralized and permissioned framework.

**PROJECT FLOW:**

****

**ADVANTAGES:**

**1. Enhanced Security:** The integration of a blockchain-based access control and authentication model adds an extra layer of security to IPFS file storage. Blockchain's inherent tamper-resistant nature and cryptographic principles contribute to robust protection against unauthorized access and data tampering.

**2. Decentralization**: Leveraging IPFS for file storage, coupled with blockchain technology, ensures a decentralized and distributed network. This mitigates the risk of a single point of failure, enhances reliability, and reduces the susceptibility to cyber attacks.

**3. Granular Access Control:** The proposed model allows for dynamic modification of access control lists through smart contracts. This granular control empowers users to manage permissions efficiently, granting or revoking access as needed, providing flexibility and adaptability in data sharing scenarios.

**SOFTWARE FRONT END REQUIREMENTS**

# **H/W CONFIGURATION:**

# Processor - I3/Intel Processor

Hard Disk - 160GB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

Monitor - SVGA

RAM - 8GB

**S/W CONFIGURATION:**

* Operating System : Windows 7/8/10
* Server side Script : HTML, CSS, Bootstrap & JS
* Programming Language : Python
* Libraries : Flask, Pandas, Mysql.connector, Os, Smtplib, Numpy
* IDE/Workbench : PyCharm
* Technology : Python 3.6+
* Server Deployment : Xampp Server

**MODULES/IMPLEMENTATION**

**DATA OWNER:**

**Register:** The data owner will register with name, email, password, mobile number, address and profile.

**Login:** After successful login the data owner can login with valid credentials, like email and password.

**Upload file:** The data owner can share file to particular file requested user.

**Generate key:** For security purpose the data owner will share key to user.

**Logout:** Data owner should be logout.

**USER MODULE:**

**Register:** The data user will register with name, email, password, mobile number, address and profile.

**Login:** After successful login the data user can login with valid credentials, like email and password.

**Send request file:** The data user needs to send a request to data owner for data.

**Enter key to download:** The user needs to enter key sent by data owners to download file.

**View storage overhead graph:**  The data user can view the output by graphical representation.

**Logout:** Data user should be logout.