AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
Peer Reviewed, Referred & Indexed Journal
E-ISSN:3069-0102
VOL.6, NO. 2(2026)

A Hybrid IPFS - Ethereum Framework for Decentralized Cloud Storage Security

K. Vijaya Bhaskar Reddy[1], Kethavath Vinod[2], Sheri Shiva Prasad[2], Machini Manikanta[2]

[1]Assistant Professor, [2]UG Student, [1,2]Department of Computer Science and Engineering

[1,2]Kommuri Pratap Reddy Institute of Technology, Ghanpur, Ghatkesar, 501301, Telangana, India.

*Correspondence:K. Vijaya Bhaskar Reddy (vbrkonda7@gmail.com)

ABSTRACT

The rapid growth of cloud computing has significantly transformed the way digital data is stored, managed, and accessed, enabling efficient sharing of information across distributed environments; however, this advancement introduces serious concerns related to data security, privacy, and trust, particularly when sensitive information is involved. Many cloud-based systems rely on centralized architectures, which increase the risk of single points of failure, unauthorized access, data tampering, and limited transparency in tracking data activities. These systems often depend on basic encryption techniques without strong auditing or verification mechanisms, making them vulnerable to insider misuse and external cyber threats, while the absence of immutable records reduces accountability and weakens user confidence. Such limitations highlight the need for a secure and transparent data-sharing framework that ensures confidentiality, integrity, and traceability of data transactions. To address these challenges, the proposed system introduces a secure cloud data sharing model developed using the Django framework, integrating Elliptic Curve Cryptography (ECC), blockchain technology, and the InterPlanetary File System (IPFS) to enhance data protection and decentralization. In this approach, ECC is used to generate cryptographic keys and encrypt user files before storage, ensuring that only authorized users can decrypt the data using the corresponding private key. The encrypted files are stored using IPFS for distributed and content-addressable storage, while file metadata such as username, filename, and timestamp is recorded on the blockchain through smart contracts using Web3, providing a decentralized and tamper-resistant record of all transactions. The system also supports user registration, authentication, secure file upload, and controlled file download functionalities, ensuring secure access and traceability. During file access, encrypted data is retrieved from IPFS and decrypted using ECC to maintain end-to-end security. By combining cryptographic encryption, decentralized storage, and blockchain-based verification, the proposed system enhances data confidentiality, prevents unauthorized modifications, and establishes a reliable and transparent framework for secure cloud data sharing.

Key words: Data Security, Data Privacy, Blockchain, Smart Contracts, Elliptic Curve Cryptography (ECC), Secure Cloud Data Sharing.

1. INTRODUCTION

The rapid growth of digital technologies has led to an exponential increase in data generation, making secure and efficient storage a critical requirement in modern computing environments. Cloud computing has emerged as a widely adopted solution due to its scalability, flexibility, and ease of access; however, it also introduces significant concerns related to data security, privacy, and trust. Sensitive information stored in cloud environments is often exposed to risks such as unauthorized access, data breaches, and manipulation, primarily due to the reliance on centralized architectures.

Blockchain technology has gained attention as a potential approach to address these concerns by providing a decentralized and immutable ledger that enhances data reliability and security. Through the use of cryptographic techniques and tamper-resistant transaction records, blockchain ensures data integrity, authenticity, and traceability, thereby reducing vulnerabilities associated with conventional data storage systems [1, 2].

Despite these advantages, ensuring robust data protection in cloud environments remains a challenging task [3]. Trusted computing systems are often employed to process sensitive information by combining hardware and software-based security mechanisms that meet predefined reliability standards. However, the increasing number of security breaches highlights the limitations of such approaches, particularly in centralized infrastructures. Typically, sensitive data is stored in encrypted form, and access is controlled through external key management systems or third-party authorities. This dependency introduces additional risks, as any compromise or lack of trust in these entities can weaken the overall security framework and expose data to potential threats.

The limitations of centralized systems and third-party dependencies have driven interest in decentralized approaches for secure data management. Blockchain technology offers key features such as decentralization, transparency, autonomy, and cryptographic security, which make it a more reliable alternative to traditional storage methods. Its ability to maintain immutable records and enable distributed verification reduces reliance on trusted intermediaries while enhancing accountability and trust. As a result, the integration of blockchain concepts into data storage and sharing environments is being explored as a means to strengthen security, improve transparency, and address the evolving challenges of modern data protection [4, 5].

Problem Definition

Centralized cloud storage systems suffer from critical security and trust issues, including single points of failure, unauthorized data access, and lack of tamper-proof audit mechanisms. Data owners have limited control over stored information and cannot independently verify data integrity. Existing systems also struggle with transparent access tracking and secure sharing of sensitive files. These limitations create significant risks for organizations handling confidential and mission-critical data.

2. LITERATURE SURVEY

Saini, et al. [6] proposed a smart contract-based access control framework for cloud-enabled smart healthcare systems. The architecture leveraged blockchain to decentralize access management, where multiple smart contracts handled identity verification, access authorization, and auditing of medical data usage. The system eliminated dependence on centralized authorities by enforcing access policies through immutable smart contract logic. It also maintained tamper-proof logs of access requests, improving accountability and traceability. Experimental evaluation demonstrated that the framework achieved low computational overhead while ensuring strong security, making it suitable for real-time EMR access scenarios. Srinivasa Naresh, et al. [7] developed a blockchain-based patient-centric healthcare communication system that enhanced secure data exchange among healthcare stakeholders. The framework enabled patients to act as primary owners of their medical data, allowing them to grant or revoke access permissions dynamically. Blockchain was used to securely record communication events, ensuring data integrity and preventing unauthorized modifications. By removing centralized intermediaries, the system reduced security risks and improved trust. The model also supported interoperability across different healthcare systems, enabling efficient and secure information sharing. Ding, et al. [8] introduced “Bloccess,” a blockchain-based framework designed to achieve fine-grained access control in distributed and untrusted environments. The system adopted a permissioned blockchain model to store and enforce access policies in a decentralized manner.

Users were allowed to define customizable access rules, which were executed through smart contracts, enabling flexible and dynamic policy management. The framework followed a user-centric approach, ensuring that data owners retained full control over access permissions. Security analysis demonstrated that the system effectively mitigated unauthorized access and policy manipulation threats. Wang, et al. [9] proposed a hybrid architecture that combined blockchain with edge computing to enable secure data aggregation in IoT-based systems. The model distributed computational tasks to edge nodes, reducing latency and bandwidth usage while supporting real-time data processing. Blockchain was utilized to validate and store aggregated data, ensuring tamper resistance and trust among distributed entities. The framework incorporated encryption and consensus mechanisms to secure data transmission and storage. Performance evaluation showed that the system improved scalability and efficiency compared to traditional centralized approaches. Chen, et al. [9] presented a blockchain-based medical data sharing mechanism that incorporated attribute-based access control to enforce fine-grained data privacy. The system allowed data owners to define access policies based on user attributes, ensuring that only authorized users could access specific data. Blockchain was used to store access policies and transaction records, providing transparency and preventing unauthorized modifications. Privacy-preserving techniques were also integrated to protect sensitive patient information during storage and transmission. Experimental results demonstrated that the system effectively balanced data accessibility with privacy protection.

Doshi, et al. [10] proposed a blockchain-based decentralized cloud storage architecture to address the limitations of traditional centralized storage systems. The model distributed data across multiple nodes while using blockchain to maintain metadata and verify data integrity. This decentralized approach eliminated single points of failure and improved fault tolerance. Cryptographic techniques were employed to secure stored data and restrict access to authorized users only. The framework demonstrated improved data security, availability, and reliability, making it suitable for sensitive applications such as healthcare data management. Meng, et al. [11] investigated decentralized storage mechanisms based on blockchain technology to enhance data reliability, integrity, and availability. Their study focused on designing distributed storage architectures that eliminated reliance on centralized servers. Blockchain was utilized to maintain immutable records of data transactions, ensuring consistency and preventing unauthorized modifications. The authors analyzed different storage strategies and evaluated their performance in terms of scalability, security, and efficiency. The results demonstrated that decentralized storage significantly reduced vulnerabilities associated with centralized systems while maintaining reliable data management.

Jabarulla, et al. [12] proposed a blockchain-based distributed patient-centric image management system for secure storage and sharing of medical imaging data. The framework enabled patients to retain full control over their medical images by managing access permissions through blockchain mechanisms. Blockchain was used to store metadata and access logs, ensuring transparency, traceability, and immutability. The system supported secure data sharing among healthcare providers without relying on centralized storage. Experimental evaluation showed that the framework achieved high scalability and efficiency, making it suitable for handling large volumes of medical imaging data. Athanere, et al. [13] introduced a hierarchical semi-decentralized data sharing framework integrating blockchain with IPFS to improve security and efficiency. In this model, data was encrypted and stored in IPFS as distributed fragments, each identified by unique hash values. Blockchain was used to manage access control policies and maintain immutable transaction records. The hierarchical structure reduced storage overhead on the blockchain while improving scalability. The framework ensured that only authorized users could reconstruct the original data, thereby enhancing privacy and data integrity. Performance analysis demonstrated that the approach achieved efficient and secure data sharing in distributed environments.

3. PROPOSED METHODOLOGY

The system architecture is designed as a secure and decentralized cloud data sharing framework that integrates a Django-based web application with Elliptic Curve Cryptography (ECC), blockchain technology, and InterPlanetary File System (IPFS) for enhanced data protection and transparency. The architecture consists of multiple interconnected layers, including the user interface, application layer, security layer, and decentralized storage layer, which collectively ensure secure data flow and controlled access. Initially, users interact with the system through a web interface that supports registration, login, file upload, and download operations. The Django application processes HTTP requests and performs user authentication by verifying credentials stored on the blockchain, ensuring trust and eliminating reliance on centralized validation. Once authenticated, users can upload files, which are processed by the file management module to generate timestamps and prepare data for encryption. The security layer employs ECC to generate public and private keys, encrypt uploaded files using the public key, and ensure that only authorized users with the corresponding private key can decrypt the data.

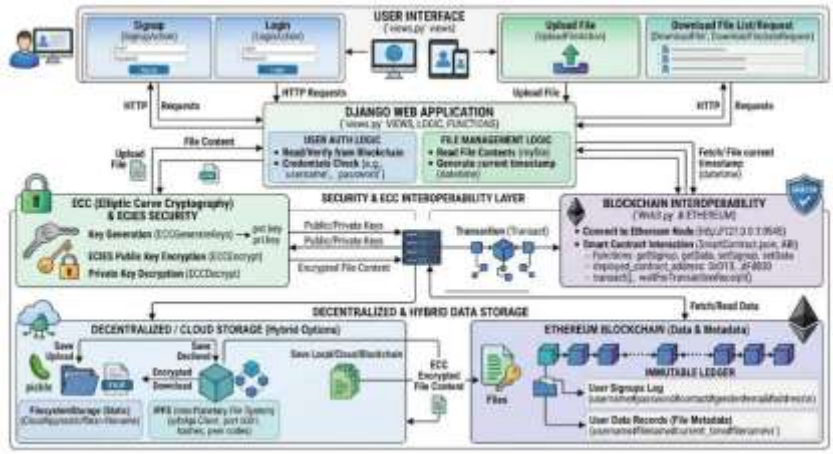

Fig. 1: System architecture of cloud storage security.

The encrypted file content is then stored in decentralized storage systems such as IPFS and local cloud storage, providing redundancy and distributed access. Simultaneously, file metadata including username, filename, and timestamp is recorded on the Ethereum blockchain through smart contracts, ensuring immutability and traceability of all transactions, as illustrated in Fig. 1. The blockchain layer maintains user registration logs and file records in an immutable ledger, preventing unauthorized modifications. During file retrieval, the system fetches metadata from the blockchain, locates the encrypted file from storage, and decrypts it using ECC before delivering it to the user. This layered architecture ensures secure communication, decentralized control, data integrity, and transparent auditing, making the system reliable and resistant to various security threats.

User Authentication and Request Handling: Users interact with the system through signup and login interfaces, where credentials are submitted via HTTP requests to the Django application. The system verifies these credentials by accessing records stored on the blockchain, ensuring tamper-proof and secure authentication. This process eliminates dependency on centralized databases and enhances trust in user validation.

File Upload and Preprocessing: After successful authentication, users upload files through the web interface, which are then received and processed by the application layer. The system generates relevant metadata such as filename and timestamp, which is essential for tracking and identification. This prepared data is then forwarded to the security layer for encryption and further processing.

ECC-Based Encryption Mechanism: The system uses ECC to generate public and private keys required for encryption and decryption processes. The uploaded file is encrypted using the public key, ensuring that the data remains confidential during storage and transmission. Only users with the corresponding private key can decrypt the file, providing strong and secure access control.

Decentralized Storage using IPFS and Cloud: The encrypted file is stored in decentralized storage systems such as IPFS along with optional local cloud storage. IPFS ensures distributed, content-addressable storage, improving availability and fault tolerance while preventing single points of failure. This approach enhances data reliability and accessibility across the network.

Blockchain-Based Metadata Management: The system records file-related metadata such as username, filename, and timestamp on the Ethereum blockchain using smart contracts. This creates an immutable and transparent ledger that securely tracks all transactions and data activities. It ensures that records cannot be altered, thereby improving data integrity and accountability.

Secure File Retrieval and Decryption: When a user requests a file, the system retrieves the corresponding metadata from the blockchain and identifies the file location in storage. The encrypted file is then fetched and decrypted using ECC with the private key. Finally, the original file is securely delivered to the user, ensuring end-to-end data protection.

3.1 Elliptic Curve Cryptography (ECC)

The ECC technique serves as the backbone of data security in this research. It provides a powerful yet lightweight encryption method to protect files before they are uploaded to the IPFS network and ensures that only authorized users can decrypt them later. ECC works on the mathematical principles of elliptic curves over finite fields, allowing strong encryption with smaller key sizes compared to traditional algorithms like RSA. In this research, ECC is responsible for generating unique key pairs for each user a public key for encryption and a private key for decryption. Before any file is stored on IPFS, it is encrypted using the user's public key, ensuring confidentiality even in a distributed environment as illustrated in Fig. 2. When a file needs to be retrieved, the user's private key is used to decrypt it back into its original form. This cryptographic approach ensures end-to-end security, data authenticity, and user-specific access control across the decentralized storage framework.

Key Generation Process: The ECC system begins by generating a pair of cryptographic keys one public and one private. The private key is a randomly selected large number, while the public key is derived mathematically from it using an elliptic curve function. These keys are unique to each user and form the basis for secure encryption and decryption processes.

Private Key Storage and Protection: Once generated, the private key is securely stored on the user's local machine or within the Django framework's encrypted storage. This key is never transmitted or shared over the network to avoid unauthorized access. It acts as the ultimate decryption key that enables users to unlock their encrypted files safely.

Public Key Distribution: The public key, unlike the private key, can be freely shared within the system. It is stored in the blockchain-based user record during registration for verification and encryption purposes. When other entities or the system needs to encrypt data for that user, they use the associated public key.

File Encryption Using Public Key: Before a file is uploaded to IPFS, it is first encrypted using the user's public key. This converts the plain data into an unreadable cipher format that only the corresponding private key can decrypt. Even if the encrypted file is accessed on the IPFS network, it remains useless to unauthorized users.

Encryption Process Implementation: The encryption process uses an ECC function that transforms the file data into a series of encrypted points on the elliptic curve. Each byte or data block is mathematically mapped using the public key curve parameters. This ensures that the encrypted data cannot be reversed without the private key's knowledge.

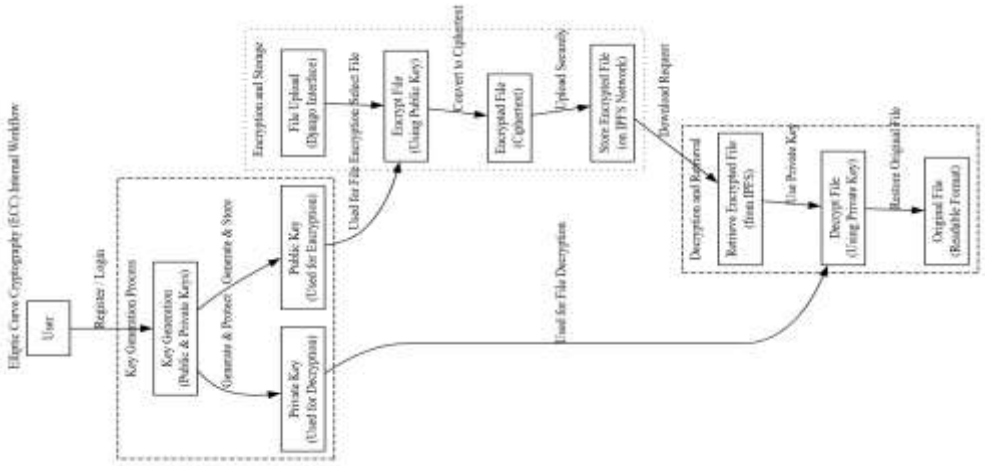

Fig. 2: Elliptic curve cryptography of cloud storage security.

Data Transmission to IPFS: Once the encryption is complete, the encrypted file is transmitted to the IPFS network for decentralized storage. Even during transmission, the encrypted nature of the file ensures it cannot be intercepted or tampered with. This maintains data confidentiality throughout the file's journey to distributed nodes.

File Retrieval and Ciphertext Access: When the user requests to download a file, the system retrieves the encrypted version from IPFS. The file remains in cipher form during the entire retrieval process, preventing unauthorized reading. This ensures that even if intermediate network nodes access the data, they cannot interpret it.

File Decryption Using Private Key: The user's private key is used to decrypt the encrypted data back into its original readable form. The decryption process mathematically reverses the elliptic curve operations applied during encryption. This restores the file exactly as it was before encryption, ensuring accuracy and data integrity.

Data Integrity Verification: After decryption, the system verifies the file's integrity using its hash value stored in the blockchain. If the computed hash matches the stored one, it confirms that the file has not been altered or corrupted. This step guarantees authenticity and reinforces trust in the entire data handling process.

4. IMPLEMENTATION DESCRIPTION

The implementation of this research integrates blockchain technology, elliptic curve cryptography, and web-based application development to create a secure, decentralized, and transparent cloud storage system. The backend is developed using Django, while Web3.py facilitates interaction with the Ethereum blockchain, ensuring all user and file-related transactions are immutable. Encryption and decryption of files are managed through ECIES to protect data confidentiality during storage and transfer.

System Initialization: When the application starts, Django initializes the web framework to handle HTTP requests from users. The research defines routes for different functionalities such as signup, login, file upload, and file download. The global variables (details, username) are used to maintain user context during active sessions.

ECC Key Generation Module: The function ECCGenerateKeys() manages public and private key generation required for cryptographic operations.

- If keys already exist (pvt.key, pri.key), they are loaded from disk.
- Otherwise, the system generates new keys using the ecies.utils.generate_eth_key() function.
- These keys are stored securely and are reused for encrypting and decrypting user files.

ECC Encryption and Decryption: Two key functions handle file security:

- Encrypts uploaded files using the ECC public key to convert plain binary data into secure ciphertext.
- Decrypts encrypted data using the private key to retrieve the original file content.

Blockchain Integration: The blockchain acts as the secure ledger for storing user credentials and file metadata.

- The readDetails() function connects to the Ethereum node (127.0.0.1:9545) via Web3.py, loads the deployed smart contract using the ABI from SmartContract.json, and retrieves records using functions like getSignup() and getData().
- The saveDataBlockChain() function writes new data into the blockchain by invoking smart contract methods (setSignup() and setData()), ensuring immutability and transparency.

User Authentication Process: The SignupAction() and LoginAction() functions manage user registration and login:

- During signup, the system verifies whether the username already exists by fetching blockchain data using readDetails('signup').
- If unique, user details are concatenated and stored on the blockchain via saveDataBlockChain().
- During login, credentials are compared with blockchain records, allowing secure verification without relying on centralized databases.

File Upload and Storage: The UploadFileAction() function implements the secure upload process:

1. The user selects a file via the web interface.
2. The file is read and encrypted using ECC.
3. The encrypted data is saved to the local directory CloudApp/static/files/.
4. Metadata (username, file name, timestamp) is sent to the blockchain using saveDataBlockChain(data, "userdata").

File Download and Decryption: The DownloadFileDataRequest() function enables users to securely download their files:

- The encrypted file is retrieved from the storage directory.
- The ECC private key decrypts the file content.

- The decrypted data is then provided to the user as a downloadable response.

Blockchain-based Data Retrieval: The Download File() function displays all user-uploaded files by fetching blockchain records using readDetails('userdata'). It dynamically generates an HTML table containing file owner, IPFS hash (or placeholder), upload timestamp, and download links. This guarantees that all displayed information is authentic, tamper-proof, and traceable via blockchain verification.

Web Interface Implementation: The Django web framework connects all backend operations with user-friendly HTML templates:

- index.html: Home page
- Login.html: User authentication
- Signup.html: New user registration
- UploadFile.html: File upload portal
- ViewSharedMessages.html: File download and viewing interface

5. CONCLUSION

This study presents a hybrid framework that combines IPFS and Ethereum blockchain to enhance security in decentralized cloud storage environments. The approach integrates cryptographic mechanisms, distributed storage techniques, and blockchain-based metadata handling to provide a reliable and secure data management solution. Data confidentiality is maintained using elliptic curve-based encryption, ensuring that sensitive information remains protected from unauthorized access. At the same time, IPFS enables content-addressable and distributed storage, which minimizes reliance on centralized servers and improves data availability. The incorporation of blockchain technology for storing metadata and access logs ensures immutability and transparency, allowing all transactions to be securely tracked and verified. System efficiency is improved by storing large files in IPFS rather than directly on the blockchain, which significantly reduces storage burden and transaction costs. Furthermore, the use of content-based addressing facilitates faster data retrieval and eliminates duplicate storage, thereby enhancing overall system scalability and performance.

References

[1] Q. Lyu, Y. Qi, X. Zhang, H. Liu, Q. Wang, N. Zheng SBAC: A secure blockchain-based access control framework for information-centric networking J. Netw. Comput. Appl., 149 (2020), Article 102444

[2] Gai et al., 2020 K. Gai, J. Guo, L. Zhu, S. Yu Blockchain meets cloud computing: A survey IEEE Commun. Surv. Tutorials, 22 (2020), pp. 2009-2030

[3] Bhari, S.; Quraishi, S.J. Blockchain and Cloud Computing-A Review. In Proceedings of the 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing, COM-IT-CON 2022, Faridabad, India, 26–27 May 2022.

[4] Taherdoost, H. Blockchain and Healthcare: A Critical Analysis of Progress and Challenges in the Last Five Years. *Blockchains* 2023, *1*, 73–89.

[5] Atieh, A.T. The Next Generation Cloud technologies: A Review On Distributed Cloud, Fog And Edge Computing and Their Opportunities and Challenges. *Res. Rev. Sci. Technol.* 2021, *1*, 1–15. Available online: https://researchberg.com/

[6] Saini, Akanksha & Qingyi, Zhu & Singh, Navneet & Xiang, Yong & Gao, Longxiang & Zhang, Yushu. (2020). A Smart Contract Based Access Control Framework for Cloud Smart Healthcare System. IEEE Internet of Things Journal. PP. 10.1109/JIOT.2020.3032997.

[7] Srinivasa Naresh, Vankamamidi & Reddi, Sivaranjani & Allavarpu, Vvl. (2021). Blockchain-based patient centric health care communication system. International Journal of Communication Systems. 34. 10.1002/dac.4749.

[8] Ding, Yepeng & Sato, Hiroyuki. (2020). Bloccess: Towards Fine-Grained Access Control Using Blockchain in a Distributed Untrustworthy Environment. 10.1109/MobileCloud48802.2020.00011.

[9] Wang, X.; Garg, S.; Lin, H.; Kaddoum, G.; Hu, J.; Hossain, M.S. A Secure Data Aggregation Strategy in Edge Computing and Blockchain-Empowered Internet of Things. *IEEE Internet Things J.* 2020, *7*, 14237–14246. Chen, Yingwen & Meng, Linghang & Zhou, Huan & Xue, Guangtao. (2021). A Blockchain-Based Medical Data Sharing Mechanism with Attribute-Based Access Control and Privacy Protection. Wireless Communications and Mobile Computing. 2021. 1-12. 10.1155/2021/6685762.

[10] Doshi, D., Khara, S. (2021). Blockchain-Based Decentralized Cloud Storage. In: Raj, J.S. (eds) International Conference on Mobile Computing and Sustainable Informatics .ICMCSI 2020. EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-030-49795-8_54

[11] Meng, L.; Sun, B. Research on Decentralized Storage Based on a Blockchain. Sustainability 2022, 14, 13060. https://doi.org/10.3390/su142013060

[12] Jabarulla, M.Y.; Lee, H.-N. Blockchain-Based Distributed Patient-Centric Image Management System. Appl. Sci. 2021, 11, 196. https://doi.org/10.3390/app11010196

[13] Smita Athanere, Ramesh Thakur, Blockchain based hierarchical semi-decentralized approach using IPFS for secure and efficient data sharing, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 4, 2022, Pages 1523-1534, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2022.01.019.