
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2025 SCPE. Volume 26, Issues 2, pp. 517–530, DOI 10.12694/scpe.v26i2.3942

ENHANCING CLOUD DATA SECURITY THROUGH AN ENCRYPTED AND EFFICIENT CONNECTION MODEL BASED ON BLOCKCHAIN TECHNOLOGY

SONALI SHARMA*, SHILPI SHARMA†, AND TANUPRIYA CHOUDHURY ‡

Abstract. Centralized file storage systems are widely used for sharing data and services, offering oversight capabilities to governing bodies. Despite their prevalence, these systems raise significant concerns about potential data misuse by authorities and control over internet information dissemination. The vulnerability of centralized systems has been highlighted by security breaches that exposed user data. To counteract these issues, researchers have proposed blockchain-based solutions for enhanced data protection. This research work highlights an innovative approach utilizing the services of Interplanetary File System (IPFS), a secure and encrypted connection framework accessible via web browsers that supports global data storage and distribution on a decentralized peer-to-peer network. This model ensures privacy, accuracy of information, and eliminates additional service fees or software requirements. However, even decentralized networks like IPFS have faced cyber threats compromising user data. In response to this challenge, the presented research outlines a model incorporating AES-256 encryption to protect files before uploading them onto IPFS nodes. Secure transmission is achieved through WebRTC technology which facilitates the exchange of encrypted file hashes and keys with recipients. The key takeaways from this study include an optimized method for file storage and distribution designed to thwart cyber-attacks targeting stored or shared files while also providing economic benefits by removing extra storage costs associated with large files in centralized systems. Overall, the research offers a robust solution aimed at safeguarding user privacy without sacrificing functionality or incurring high costs.

Key words: IPFS, AES-256, Web RTC, Blockchain Technology, Decentralized, Distributed

1. Introduction.

The current version internet comprises of the widespread usage of services provided by the Web2 environment. The online transactions including- sharing of data, communication of information, confidential file storage and end to end transfer of content is governed by the centralized systems. The significant drawbacks of accelerating transactions of data and information through the Web2 environment include- services in exchange of personal data of end users, centralized data monitoring by the governing bodies where the end users often mistake themselves to be the owner of shared information and limited control of the metadata produced on the Web2 environment [1]. Web3 is the new generation of internet which attracts attention due to its decentralization abilities. This also means that the privileges of artificial intelligence and cyber security can be implemented to make the internet more secure and empower the individuals to have complete control over the content they create or share online [2]. Web3 is often termed as semantic or spatial web that leverages decentralization capabilities. A distinct concept from Web 3 which focuses on capabilities of 3D virtual realm and augmented reality is termed as Metaverse. Metaverse is an environment wherein the three dimensional objects interact and provide better interface for human engagement. Therefore, Web3 and Metaverse can be used together for providing a more interactive environment for the users but they are not interchangeable [2]. Decentralized applications (DApp) that operate on the concept blockchain technology and other distributed platforms play a crucial role in making the Web3 environment permission less, secure and private. Under the proposed approach, the files are uploaded on the web using the potential of the Web3 environment and are distributed using interplanetary file system (IPFS) to ensure security and ownership of the content by the creators. However, the modern day malicious software including the zero day attacks can infiltrate any device. To safeguard our system from the viruses, we can scan the files before uploading/downloading them by any efficient anti-virus software available. However, the malicious files cannot damage the file storage system

*Amity University, Uttar Pradesh, India (sonali.260893@gmail.com).
†Department of computer Science and Engineering, Amity University, Uttar Pradesh, India (ssharma22@amity.edu).
‡School of Computer Sciences, University of Petroleum and Energy Studies (UPES), Dehradun, Uttarakhand, India (tanupriya@ddn.upes.ac.in).

implemented using IPFS but can potentially harm any user system who downloads such files from IPFS. Our research aims to propose a model that can facilitate the transition between the Web2 and Web3 seamlessly.

1.1. Contribution. Cloud based platforms are crucial for files and data storage. However, they raise security concerns which impact the integrity and confidentiality of the data shared on the centralised platforms. They are expensive and susceptible to man in the middle attacks. The proposed and implemented model presented in this paper ensures data security of the stored and shared data over the network. To achieve the server less architecture we have designed a user interactive framework using React and Next.js. To establish a secure connection between the sender and receiver WebRTC has been used and decentralisation is achieved using IPFS. The framework incorporates the data sharing mechanism using AES-256 file encryption method. The research paper presents two novel algorithms, which have been implemented to ensure data confidentiality and integrity. The proposed framework has been compared with the existing frameworks to demonstrate its efficiency and cost-effectiveness.

The research paper is arranged as Literature Review in section 2, Motivation in section 3, Methodology in section 4, Proposed Framework in section 5, Implementation of Framework in section 6, Results and conclusion in section 7 and Future Work in section 8.

2. Literature Review. The centralised systems work on the capabilities of the client-server approach powered by HTTP (Hyper Text Transfer Protocol) due to its well implemented architecture to meet the industry standards for communication of data. Here are some drawbacks of accessing data and information using the potential of the HTTP architecture.

Limited Bandwidth: Channelising information through single server to multiple clients.

Paid services: Global access to the shared data requires a server to be setup or the services provided by the central storage application is paid.

Ephemeral: Failure of the centralized server, will lead to data loss.

Data Duplication: Files having similar information, name and metadata can be found at multiple servers.

IPFS on the other hand provides better access to the data by decentralizing it, so that even if one node breaks down, the data must be present with many other nodes. IPFS has high bandwidth as the data is not downloaded from single server, it is accessed by many nodes having that information [3]. Similar to the centralized data storage mechanisms IPFS also maintains a cache or a list of frequently accessed files, if the user wants a file to be available for quick access then cryptocurrencies can be used to incentivize the information or there is a cost free mechanism of file pinning which allows quick access to the frequently requested files. Using the mentioned method the file will be consistent and available in a network whenever required. To prevent data redundancy IPFS uses the concept of content based addressing where the files with same name and data cannot be stored even on different locations [4]. It is important to note that both HTTP and IPFS protocols provide data sharing and accessibility mechanisms and but have very distinct structure:

Content addressing: HTTP locates the stored information by the help of URL (Uniform Resource Locator), whereas IPFS uses content-based addressing for locating the stored resources through the cryptographic hash of the content regardless of its location.

Decentralization: HTTP relies on centralized servers to host and distribute content, while IPFS is a decentralized network where content is distributed across many different nodes [5]. This means that IPFS can be more resilient to network failures and censorship.

Performance: HTTP typically relies on a client-server architecture, where a client requests data from a server and the server responds with the requested data [5].

Persistence: To gather information through HTTP architecture, it relies on the centralized server up time and as long as it is available for data retrieval. Whereas, IPFS structure depends on the availability of nodes hosting the data on network.

While HTTP and IPFS both serve the purpose of sharing and accessing content over the internet, IPFS offers some unique advantages over HTTP, particularly in terms of decentralization, content addressing, and persistence [6]. However, HTTP is still widely used and has a well-established infrastructure that many websites and applications rely on. Fig. 2.1 shows average analysis on the research work published worldwide from the year 2018 to 2024 on IPFS protocol. The analysis has been done using VOSviewer with input data from the

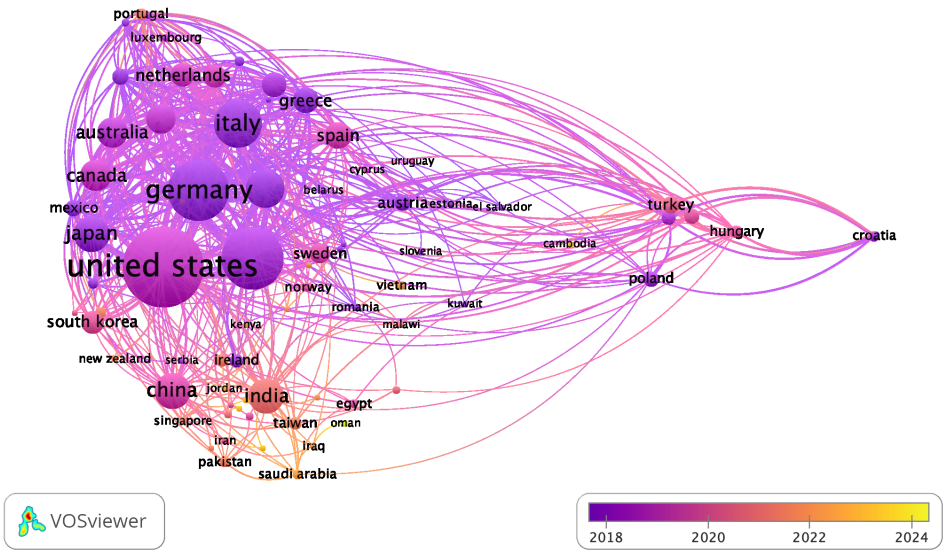

Fig. 2.1: Analysis of average documents published from 2018 to 2024 on the study of IPFS

OpenAlex API. The results showcase links for 3430 documents published worldwide. The maximum research work has been initiated in the United States followed by United Kingdom and Germany.

Service availability is one of the major concerns when we implement distributed/decentralised framework and mechanisms. Traditional methods of calculating the uptime of service is measured by percentage of available time, explained by the below mentioned equation [7][8].

$$Availability = \frac{MTTF}{MTTF + MTTR} \tag{2.1}$$

MTTF: Mean Time Failure of service
MTTR: Mean Time to Repair the service

However, the above equation does not explain the reason of service failure. In case of IPFS or decentralised frameworks, there can be many reasons for service failure like downtime of requested node, network congestion, and unavailability of requested of data. To better describe the data or service availability, Giwon On, et al. [7, 8] have mentioned an optimised definition of service and data availability of IPFS:

$$Avail_{Service} = Avail_{Data} X Avail_{System} \tag{2.2}$$

$$Avail_{System} = Avail_{Node} X Avail_{Link} \tag{2.3}$$

$$Avail_{Node} = Avail_{Dynamic} X Avail_{Intrinsics} \tag{2.4}$$

We elaborate the availability matrix on the basis of our proposed framework using IPFS.

$Avail_{Service}$**:** The available service of our proposed framework is directly proportional to data availability and system availability. Therefore, service completion depends on available data and system uptime.

$Avail_{Data}$**:** This refers to availability of updated data for user access and use. IPNS (Interplanetary Name System) and dynamic links explained in further sections are implemented in IPFS to ensure updated data availability.

Table 3.1: BitTorrent vs IPFS

BitTorrent	**IPFS**
BitTorrent does not have the capabilities of using DHT (Distributed Hash Table), therefore a torrent and meta-data file is required to start downloading the file.	Hash of the file is required to start downloading the file from IPFS. This is due to Kademlia- a lookup mechanism by IPFS which supports DHT.
There is no mechanism of leveraging the CID (Content Identifier) and DHT of the file content. Therefore, the content of the files cannot be compared while it is being uploaded from different sources. This leads to data redundancy.	Files having same CID will be detected as soon as they are uploaded on IPFS. Files having same metadata, content, file name, byte level compression and encryption will not be uploaded on IPFS due to their similar CID.

$Avail_{System}$**:** The system availability depends on availability of dynamic IPFS nodes, their configuration, and data-link availability.

$Avail_{Link}$**:** Available link means that the node link between the user and server is within the reachable limit with considerable delay to provide service using high bandwidth resource.

$Avail_{Node}$**:** This indicates the service node strength, bandwidth and performance to support the service completion and data delivery.

$Avail_{Dynamic}$**:** Stability in node uptime is dynamic node availability, to help and process the request within a tolerable time limit.

$Avail_{Intrinsics}$**:** Node Intrinsic availability refers to the bandwidth, storage capacity, system configuration and processing power of the IPFS nodes in the distributed framework.

WebRTC is based on open standards consisting of several APIs (Application Programming Interfaces) to enable standalone structure for video and audio calls, chat, and file sharing mechanism so that the developers can use it to build real-time communication applications. For fast, secure and efficient communication it uses peer-to-peer connections between the devices allowing the usage of advanced features such as encryption, which ensure private connection [9]. It therefore consulates as the building block of decentralized applications based on real-time connections. Due to the persistence and resilience of WebRTC structure there is no single point of failure, which enables its adaptability to the changing modern standards [10]. To enable low latency communication: WebSockets or signalling servers powered by WebRTC ensure device discovery and connection. Upon successful connection establishment, the information can be easily exchanged without the centralised servers. Overall, WebRTC is a powerful technology to ensure real-time communication for decentralized environment.

Another aspect of conflict is to understand the difference between IPFS and blockchain. They both serve the purpose of decentralisation with different use cases. IPFS however, showcases capabilities to overcome the limitations of blockchain [11]. One of the major limitations of blockchain include scalability, where the size of network may affect the synchronization between the blockchain nodes resulting in slow processing time and higher price. To overcome this issue IPFS works on the mechanism of storing the large files off-chain. This reduces the size of network making it more manageable, easy to verify and scalable [12].

IPFS enables the developers to create applications where blockchain based solutions to store large data can be provided without any additional cost as storing large amounts of data on-chain can be expensive. In summary, the limitations of cost, decentralization, scalability can be overcome by using the capabilities of IPFS [13].

3. Motivation. In the era of innovation, the decentralization mechanisms have led to invention of many file sharing protocols. One such popular file sharing mechanism is BitTorrent which enables the users to share the files through peer-to-peer mechanism over the internet. A group of swans (hosts) is used to download the information through BitTorrent instead of centralized servers [10]. Table3.1 shows differences between the functionalities of BitTorrent and IPFS. The file accessibility speed is similar to IPFS, where the most requested/visited file is available for faster access or download by the users.

BitTorrent and IPFS are based on data security while sharing between the users, although they do not use content based encryption. Additionally, the Content Identifier(CID) can be accessed in public domain due

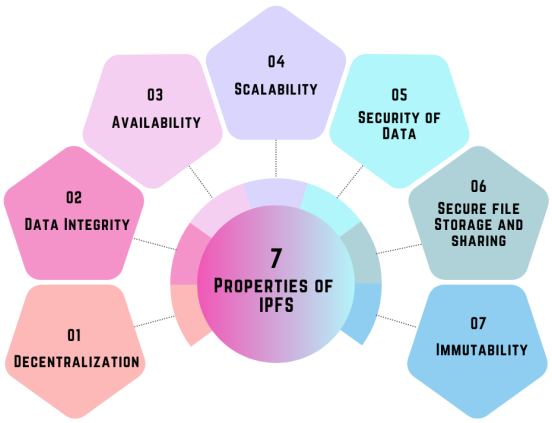

Fig. 3.1: Properties of IPFS

to the usage of public Distributed Hash Tables by IPFS [14]. Similarly, the IP addresses are public in the Bit Torrent architecture. This affects the confidentiality of the data as the hackers can constantly monitor the sources and destination of file sharing. To overcome this Filecoin, SiaCoin, SafeCoin, Internxt and Storj along with many other DApp (Decentralized applications) are recently being used as peer to peer file sharing mechanisms which aim at addressing the issues of BitTorrent and IPFS.

IPFS being an extension to the traditional and popular HTTP protocol is efficient with high bandwidth and no single point of failure. Fig. 3.1shows the properties and advantages of using IPFS in our implemented framework for ensuring security and integrity of data. Our proposed framework does not use the capabilities of an Ethereum blockchain to implement the framework using the peer to peer network since data storage is expensive.

4. Methodology. Interplanetary File system, widely known as IPFS is a content based decentralized storage protocol which can be used to store, access and retrieve files and data of any form [15]. IPFS stores the contents in two main parts:

(i) An unstructured binary data block of size 256 KB. When the file size is more than 256 KB then a list containing links of file chunks small in size are maintained.

(ii) An array having links to IPFS objects associated with the same directory.

Fig. 4.1 shows the fundamental principles of understanding the working of IPFS. Let's closely look at these features.

4.1. CID (Content Identifier). CID is a hash unique to the content of the file. Generally used in the URL to access the file by its content. This also means that if the content of the file changes then the CID will also change. In some instances changing the URL every time the content of the file is updated might not be feasible, for this IPFS has a concept of mutable pointers also known as Interplanetary Name System (IPNS). In this case the address of the pointer which is pointing to a CID is shared and every time there is any update in the content of the file the CID is updated but the address of the pointer remains the same [14]. IPFS uses DHT (Distributed Hast table) for providing efficiency and to make the file retrieval process robust. DHT is responsible for the information of all the CIDs. Therefore when a CID is requested by the user, the IPFS nodes send queries to the DHT for finding the nearest nodes which can help you retrieve the requested CID [15]. The CIDs which are requested frequently are available in the DHT for a limited time and they can also be pinned if you never want them to be garbage collected. This information about which CIDs are being requested frequently and by which nodes is publicly available.

4.2. DAG (Directed Acyclic Graph). IPFS uses the advantages of DAG data structure to split the files into blocks. All files are broken down into blocks which can be later retrieved from different sources, each

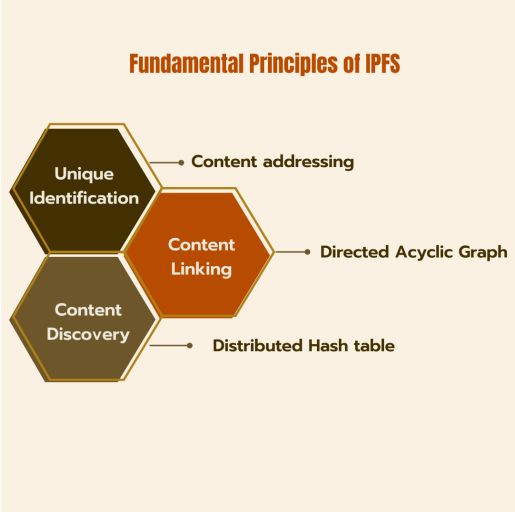

Fig. 4.1: Fundamental Principles of IPFS

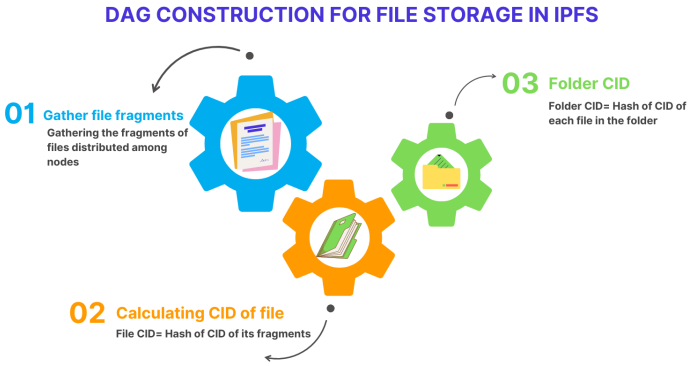

Fig. 4.2: DAG construction for file storage in IPFS

block receives a unique CID based on its content as discussed in the previous section. The CID of the file is a hash of the CID's of the blocks. Therefore, a folder containing many files has a CID which will be made by the hashes of the CID's of the files stored inside it [16]. Fig. 4.2 explains the working of the DAG.

The advantages of using DAG is that even if there are similar files in a system then different DAG's can reference to same subset of data [17]. This can be helpful if some part of the bigger folder is being updated then all other parts can continue to refer the same destinations and only the updated file can change its reference as per the requirement.

4.3. DHT (Distributed Hash Table). A hash table is the one which maps the keys to values. A distributed hash table however is split across the network and is used to connect different peers [18]. Once the content to be requested is identified using the previous two steps, location of the content and where to find the content is taken care by the DHT. Therefore DHT is responsible for:

- Identification of the peers who have the requested content
- Location of the Content

IPFS uses a data exchange protocol called Bitswap which is responsible for fetching the content from a peer and sending it to another [19]. The designing of this protocol aims at balancing the latency and throughput

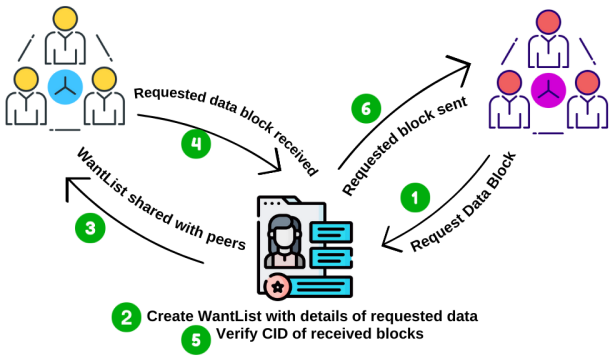

Fig. 4.3: Working of BITSWAP

of the entire system. Bitswap is a simple protocol which can receive and transfer blocks up to size 2MB for achieving maximum compatibility.

Fig 4.3 shows the working of BITSWAP protocol which helps in fetching the data without any latency real time.

If the blocks are to be retrieved from the peers then a WantList can be created which has the details of the request blocks [20]. Hash of the received contents can be verified upon retrieval of blocks from the peers using Bitswap. Verification can be done by hashing the contents to fetch CID of the received blocks and match them with the CID of the request blocks. If both match then they can be sent to the other peers who have requested the blocks.

5. Proposed Framework. IPFS is a peer to peer protocol and a globally distributed server which is used for data storage and delivery. CIDs can easily be used to reference the distributed server and the nodes that are communicating with each other. Let's closely look at the attributes of IPFS which are publicly declared to be referenced by nodes.

- Although the communication between the nodes is encrypted but the metadata produced by nodes which is shared with the DHT is public.
- DHT contains a variety of information produced by the IPFS nodes which is public- CID of data and PeerIDs (Unique node identifier).
- Retrieval and request of CIDs by the nodes.

There are two types of encryption schemes that can be used for data protection: Transport encryption and Content encryption. IPFS uses transport encryption so that anyone cannot view the files/data while they are being transported between nodes but the content is not protected as anyone who as the CID can download and view the contents of the file. The additional security measures therefore are required to be taken for data protection. The framework proposed in this paper aims at achieving privacy, confidentiality and security of data while it is shared using IPFS.

1. *Browser Application*: Build using TypeScript because it has the capabilities of specifying the type of data being passed into the code but in JavaScript this facility is not there as the variables and parameters don't give any information. The compile time type checking is possible by using the TypeScript. For user interaction Next.js and React is used so that the browser application can be easily accessed using web and no extra server storage is required to run this.
2. *Encryption of file uploaded by the user*: AES-256 is used for the encryption of files on the user system before sending the file to IPFS. AES-256 encryption is not easy to decrypt because of the large key size. It is even quantum safe because it requires 6600 logical, error-corrected qubits to break the encryption of AES-256 but the latest quantum computer Osprey has 433 qubits achieved by IBM. The computational powers of the presently used computers also cannot decrypt the files encrypted by AES-

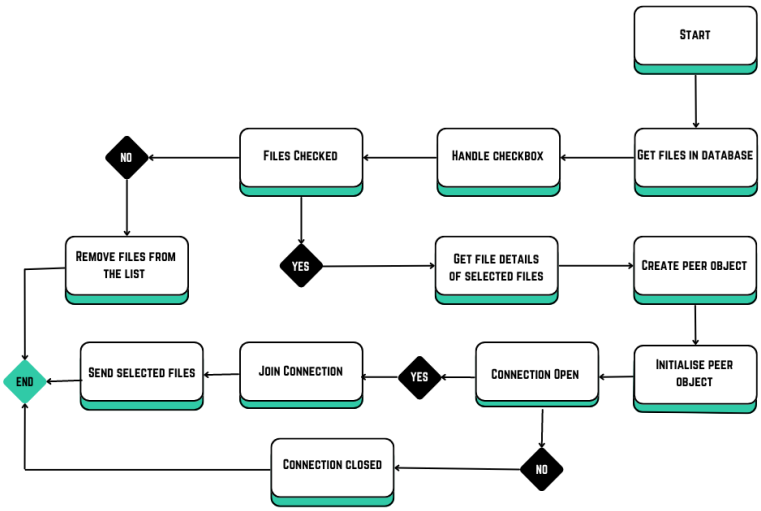

Fig. 5.1: Flowchart for file encryption and upload using the proposed framework

256 easily without the encryption key [21]. In this model the keys are being stored on the browser and a single key is used for the encryption of files [17]. Fig.5.1 shows the process of uploading the encrypted file using IPFS.

3. *Uploading encrypted files to IPFS*: For implementation and usage of IPFS environment we are using the HTTP-client library which can work with both the Go and JS implementations of IPFS without installing any extra software on the system [22]. As the files being shared with the IPFS nodes are encrypted and are being shared directly there are very remote chances of many attacks such as the man in the middle attack (MIMT)and brute force attacks. The API used for implementation of IPFS is responsible for splitting the files into blocks and create hashes of the contents. The CID is not shared with the nodes and the metadata is directly stored at the database and not in the server.
4. *File sharing*: Files uploaded to IPFS can be shared with the peers by creating and initializing a peer object. Create peer object with connection to shared peer server and initialize the peer object. Initialize an IPFS client to interact with the IPFS network. This is necessary for accessing and retrieving the uploaded files. After initialization share the data with peer object [23]. To write clear and concise instructions for sending files uploaded to IPFS to peers by creating and initializing a peer object, you can follow these steps:
   (i) Initialize a peer object to communicate with peers over a shared peer server. Make sure to provide the necessary configuration options, such as the server URL, port, and any other required settings.
   (ii) Set up callbacks to handle various events related to the peer connections, such as session timeout or data transfer events. This ensures you can react appropriately to these events.
   (iii) To share files or data with peers, you can use the connection object created in the event handler. Send the data to the connected peer using the ‘send‘ method.
   (iv) If session timeout or other connection-related events are important for your application, you can implement callbacks for these events to handle them appropriately.
   (v) If needed, you can also close connections gracefully when you’re done with them.
5. *File Retrieval:* The user request for file retrieval is processed by IPFS and information about the nodes storing the CID of the requested file is shared by referring to the DHT. After the nodes having the blocks of the requested file are identified, the encrypted file is send back along with its hash to the browser application at the user end.

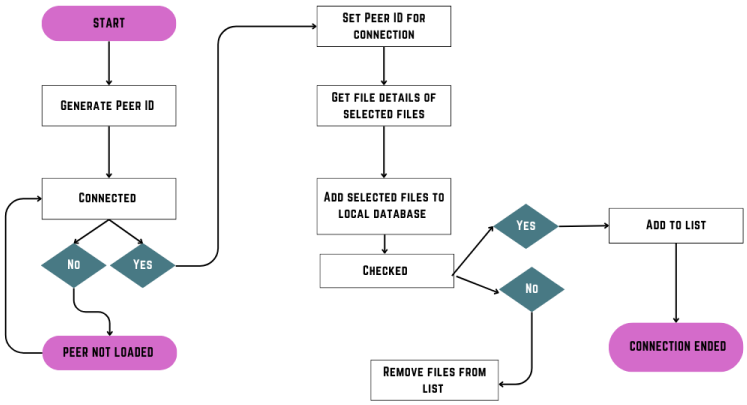

Fig. 5.2: Flowchart for file decryption and download using the proposed framework

6. WebRTC employs encryption protocols like Datagram Transport Layer Security (DTLS) and Secure Real-Time Protocol (SRTP) to ensure secure transmission of data. The user can decrypt the file using the key stored on the browser and retrieve the original readable file. Fig. 5.2 shows the retrieval process of the encrypted file.

6. Implementation of framework. The proposed framework aims to build a web based application which does not require any external server for file storage. The framework consists of the following components:
IPFS Node: A local instance of the IPFS daemon that handles file storage, retrieval, and distribution.
Encryption/Decryption Module: Encrypts files before uploading them to IPFS and decrypts them upon retrieval using a symmetric encryption algorithm.
Key Management Module: Generates, stores, and manages encryption keys.
WebRTC Signaling Server: Facilitates the exchange of signaling data between peers, enabling the establishment of a WebRTC connection.
WebRTC Data Channel: A secure communication channel for exchanging encryption keys between peers.

A user uploads a file to the system. The Encryption/Decryption module encrypts the file using a randomly generated encryption key. The encrypted file is added to the local IPFS node and distributed across the network [24]. The user shares the unique IPFS hash of the encrypted file and establishes a WebRTC connection with the recipient using the signaling server. Algorithm 1 explains the procedure to encrypt the file using a secure randomly generated password before sharing the file with the peers. The encryption key is securely transmitted to the recipient via the WebRTC Data Channel. The recipient downloads the encrypted file from IPFS using the hash and decrypts it using the received encryption key. Our proposed system ensures the following security properties:

1. *Confidentiality:* File content is encrypted using a symmetric encryption algorithm, making it unreadable to unauthorized parties.
2. *Integrity:* The content-addressed nature of IPFS ensures that the file content remains unaltered, due to a different hash value for any changes made to the uploaded file.
3. *Availability:* IPFS's distributed architecture ensures file availability even in the presence of network failures or censorship attempts.
4. *Secure Key Exchange:* WebRTC's encryption protocols (DTLS and SRTP) provide secure transmission of encryption keys, preventing eavesdropping and man-in-the-middle attacks.
5. *Cost effective:* There is no cost required for sharing the files using the proposed framework.

Table 6.1: Libraries, frameworks and technologies

Technology	Version
IPFS	Ipfs-desktop-0.24.1-mac.exe
React	18.2.0
Next.js	Next.js 13
IPFS-http-client	https://github.com/ipfs/js-ipfs.git
Web Crypto API	https://github.com/mdn/content/blob/main/files/en-us/web/api/web_crypto_api/index.md?plain=1

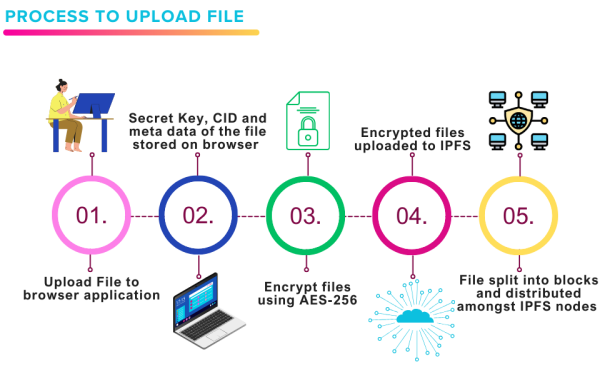

Fig. 6.1: Process diagram to upload the encrypted file using the proposed framework

6. *Low space utilization:* CID and hash are the only values stored for each file on each node, very limited space utilization is required by our system as compared to other blockchain based solutions proposed in the past.
7. *Not searchable on web:* The file is not searchable on the internet due to encrypted path information.
8. *Eavesdropping and man in the middle attack is not possible:* Unique peer ID is used for establishing the secure connection between the sender and receiver.
9. *Server less file upload/download and transportation:* No server space is required for running the proposed model.
10. *Two way security:* The established connection is verified through shared Peer ID communicated and sent using WebRTC data channel.

The technology stack used to build the web application is discussed in Table 6.1.

In the proposed web application, the files are encrypted by using AES-256 and then the encrypted file stored on the system by .enc extension can be uploaded on IPFS nodes. Fig. 6.1 and Fig.6.2 show the upload and download procedure of encrypted files using the proposed mechanism. There is no cost of managing the files as they are being uploaded on the web in a secured manner [25]. HTTPS is used for this purpose in the current searches and file storage but due to centralized client server mechanism it is considered to be prone to failure in the future. After installing the latest IPFS version on your system, for running the web application on the browser of a PC or mobile run the following commands on the terminal to start IPFS client. Algorithm 2 explains the procedure to share the encrypted files securely. The two-way security mechanism to establish the connection between sender and receiver ensures no cyber-attacks are possible while file transmission.

While establishing secure connection between the sender and receiver, a unique peer ID will be generated the similar way we generate the one time stamp to open connection for the transaction to be successful. To ensure the ease of use for the sender and receiver we have generated random IDs as a combination of adjectives + nouns + alphanumeric string. This will help in connecting and login to the connection without any delay. Additionally, this peer ID will be valid for sharing only one set of file. For another file sharing option, we have

Algorithm 1 Encrypt and upload file to IPFS

Input: IPFS API Address, File, User interaction through UI elements.
Output: Encrypted File, File CID, Stored file details in the browser.

Initialise Variables: Password, selectedFile, apiAddress, client, clientID, ipfsFile, encryptedFile, filesInDB.
Generate a password using alphanumeric characters
Create an IPFS client instance using the provided API address
while IPFS client is active **do**
  Set the client instance and client details (ID)
  for all selectedFile from input field do **do**
    Set the 'password' variable based on user input
    for encryptedFile do **do**
      Read the selected file as a Uint8Array
      Create salt, key, and IV for encryption
      Encrypt the file using AES-CBC and derived key
      $encrypteddata \leftarrow salt$
      $encryptedFile \leftarrow EncryptedData$
      Display an alert confirming successful encryption
    end for
  end for
  for all encrptedFile do **do**
    Upload the encrypted file to IPFS
    $ipfsFile \leftarrow FileCID$
    Display an alert confirming successful file upload
  end for
  if $UploadFile = PinFile$ **then**
    Pin the file to IPFS
    Display an alert confirming successful file pinning
  end if
  if ipfsFile, selectedFile, and password are available **then**
    Create a PouchDB instance
    Create a document with file details and store it in the local database
    Display an alert confirming successful file details storage
  end if
  Call getFilesInDB to refresh the file list
end while

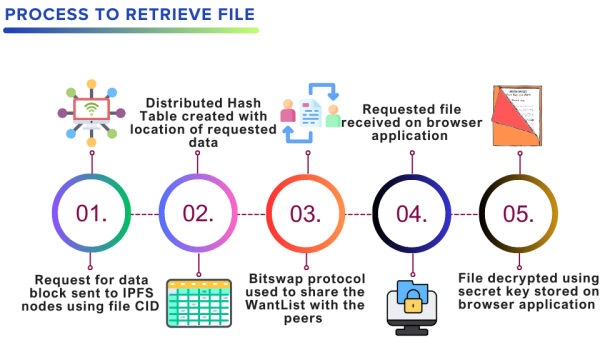

Fig. 6.2: Process diagram to download and decrypt the file from the proposed framework

to generate a different Peer ID. Due to this feature the model is secured from the brute force attacks and man in the middle attacks.

Algorithm 2 File sharing mechanism using the proposed framework

Input: Encrypted File, IPFS API address.
Output: File in database

Initialize Variables: lastPeerId, receiverId, peer, conn, status, filesInDB, selectedFiles
Create a PouchDB instance called db for the files database
Retrieve all documents from the files database
Handle Checkbox(event)
if The checkbox is checked **then**
    Add the checkbox value to the selectedFiles list
else
    Remove the checkbox value from the selectedFiles list
end if
Create an empty list selectedFilesDetails
for all Files in filesInDB **do**
    if file.id is in selectedFiles **then**
        $selectedFilesDetails \leftarrow files$
        return selectedFilesDetails
    end if
end for
Set receiverId to the value entered in the input field
Initialize a Peer Object
Create a peer object and connect to a shared PeerJS server
Create a new peer object and assign it to peer
while The peer object is successfully opened **do**
    Assign the peer ID
    Define event handlers for peer events, such as open, connection, disconnected, close, and error
    Enter the peer ID of the receiver
    Display the current connection status
    Join a connection to the destination peer based on the provided peer ID
    Select files to share
    Allow the user to refresh the list of files available in the local database
    Enable the user to select files for sharing using checkboxes
    for all Selected file do **do**
        if conn exists **then**
            Send the details of selected files to 'conn'
            Display a list of selected file CIDs that will be shared
            Send the selected file CIDs to the receiver
        else
            display an alert indicating that the connection is closed
        end if
    end for
end while
Close connection

7. Results and Conclusion. In this paper, we have presented a secure mechanism for file storage and peer-to-peer sharing which uses the decentralized capabilities of IPFS and the secure communication channel provided by WebRTC. The proposed system encrypts files before uploading them to IPFS and securely shares the encryption keys using WebRTC. Our approach provides confidentiality, integrity, and availability for shared files, while secure key exchange architecture between peers. The limitation of the proposed framework is that both the sender and receiver must have IPFS installed on their respective devices for communication of information. This can however, be made easily adaptable for use and practice as we are adapting to various new technologies for securing our file sharing systems. The robustness of the secure channel communication enables to share any size of files across the network without paying for the services like server space for storage and data encryption.

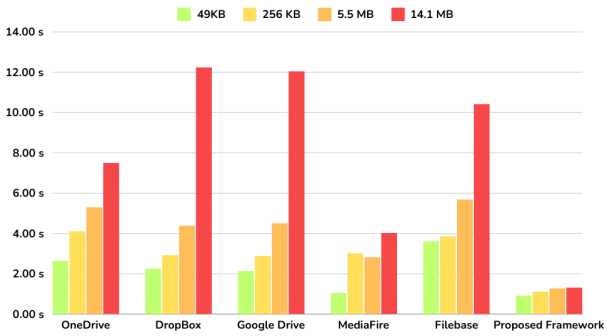

Fig. 7.1: Comparative analysis for file uploading on cloud based platforms

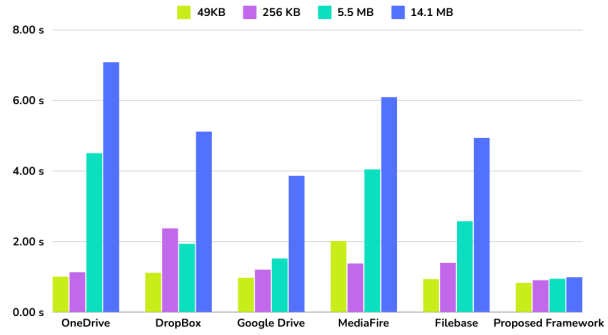

Fig. 7.2: Comparative analysis for file downloading from cloud based platforms

To compare our proposed framework with the popular cloud based solutions built on centralized server we use the React File Picker component that works with the Apideck (File Storage API). The results have been recorded based on the time parameter with 48 data points. System information – M2 MAC with 16GB RAM; tests were performed on fiber connection connected through ethernet on Safari browser. Files of different sizes have been compared for both the download and upload procedure- 49 KB, 256 KB, 5.5 MB and 14.1 MB. The proposed framework is the fastest as compared to OneDrive, DropBox, Google Drive, MediaFire and Filebase platforms as shown in Fig.7.1 and Fig.7.2.

8. Future Work. Our proposed framework combines the potential of IPFS and WebRTC to implement an encrypted, secure, decentralized peer-to-peer file sharing framework. The mechanism has the capability to serve as a foundation for further research in decentralized file sharing and content distribution. However, the further research and development can work around the following areas:

Scalability: Due to increase in the number of users, files shared across the nodes can increase and therefore its performance should be enhanced accordingly.

Access Control: These policies can be implemented to provide users with more control over the users who can access their shared files.

Key Management: Exploring more advanced key management techniques, such as key rotation, modification and regular updates to enhance security and mitigate potential key sharing risks.

Mobile Support: Mobile application for file sharing using similar technology can be implemented, which can extend the system to support mobile devices for a seamless cross-platform experience.

REFERENCES

[1] Kuai, Le, Mary Lacity, and Jeffrey K. Mullins. *Web 2 vs. Web 3 Paths to the Metaverse: Who Is Leading? Who Should Lead?*, The Journal of The British Blockchain Association, September 2023.

[2] W. C. Diehl *Artificial Intelligence, Web 3, and the Future of Distance Education*, American Journal of Distance Education, vol. 37, no. 2, pp. 83-84, 2023.

[3] C. Patsakis and F. Casino *Hydras and IPFS: a decentralised playground for malware*, International Journal of Information Security, vol. 18, pp. 787–799, 2019.

[4] S. Vimal and S. K. Srivatsa *A new cluster P2P file sharing system based on IPFS and blockchain technology*, Journal of Ambient Intelligence and Human Computing, 2019.

[5] A. A. Battah, M. M. Madine, H. Alzaabi, I. Yaqoob, K. Salah, and R. Jayaraman *Blockchain-Based Multi-Party Authorization for Accessing IPFS Encrypted Data*, IEEE Access, vol. 8, pp. 196813-196825, 2020.

[6] N. Nizamuddin, K. Salah, M. Ajmal Azad, J. Arshad, and M. H. Rehman *Decentralized document version control using Ethereum blockchain and IPFS*, Computers & Electrical Engineering, vol. 76, pp. 183-197, 2019.

[7] Zeng R., You J., Li Y., Han R. *An ICN-Based IPFS High-Availability Architecture*, Future Internet, 2022.

[8] On G., Schmitt J., Steinmetz R., *The Effectiveness of Realistic Replication Strategies on Quality of Availability for Peer-To-Peer Systems*, In Proceedings of the Third International Conference on Peer-To-Peer Computing (P2P2003),pp. 57–64, Linkoping, Sweden, 1–3 September 2003.

[9] M. Tomaiuolo, M. Mordonini, and A. Poggi *A P2P Architecture for Social Networking in Applying Integration Techniques and Methods in Distributed Systems and Technologies*, IGI Global, 2019.

[10] V. Pattanaik, I. Sharvadze, and D. Draheim *A Peer-to-Peer Data Sharing Framework for Web Browsers*, SN COMPUT. SCI., vol. 1, p. 214, 2020.

[11] W. P. scWardlaw *The RSA public key cryptosystem in Coding theory and cryptography*, Berlin: Springer, pp. 101–23, 2000.

[12] A. V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, and T. Berners-Lee *Solid: A platform for decentralized social applications based on linked data*, Tech. rep., MIT CSAIL & Qatar Computing Research Institute, 2016.

[13] A. Tenorio-Fornés, S. Hassan, and J. Pavón *Open Peer-to-Peer Systems over Blockchain and IPFS: an Agent Oriented Framework*, 2018.

[14] G. Alagic, J. Alperin-Sheriff, D. Apon, et al. *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*, NISTIR 8309, NIST, U.S. Department of Commerce, July 2020.

[15] S. El Adib and N. Raissouni *AES Encryption Algorithm Hardware Implementation Architecture: Resource and Execution Time Optimization*, International Journal of Information & Network Security (IJINS), vol. 1, no. 2, pp. 110-118, June 2012.

[16] A. K. Das *A random key establishment scheme for multi-phase deployment in large-scale distributed sensor networks*, International Journal of Information Security, vol. 11, pp. 189–211, 2012.

[17] C. Yang, T. Liang, N. Shi, B. Xu, Y. Cao, and K. Yu *AuthPrivacyChain: A Blockchain-Based Access Control Framework With Privacy Protection in Cloud*, IEEE Access, pp. 1-1, 2020.

[18] G. Alagic, J. Alperin-Sheriff, D. Apon, et al. *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*,NISTIR 8309, NIST, U.S. Department of Commerce, July 2020.

[19] X. Bonnetain, M. Naya-Plasencia, and A. Schrottenloher *Quantum Security Analysis of AES*,IACR Transactions on Symmetric Cryptology, Ruhr Universität Bochum, 2019.

[20] M. K. C. Ledda, B. D. Gerardo, and A. A. Hernandez *Enhancing IDEA Algorithm using Circular Shift and Middle Square Method*, ICT and Knowledge Engineering (ICT&KE) 2019 17th International Conference, pp. 1-6, 2019.

[21] Abbadini, M., Beretta, M., di Vimercati, S. D. C., Facchinetti, D., Foresti, S., Oldani, G., ... & Samarati, P. *Supporting Data Owner Control in IPFS Networks.*, In Proceeding of the IEEE International Conference on Communications (IEEE ICC 2024).

[22] Dwivedi SK, Amin R, Vollala S.*Smart contract and IPFS-based trustworthy secure data storage and device authentication scheme in fog computing environment.*, Peer-to-Peer Networking and Applications. 16(1):1-21; Jan 2023.

[23] Medina J, Rojas-Cessa R. *AMI-Chain: a scalable power-metering blockchain with IPFS storage for smart cities.*, Internet of Things.1:101097; Feb 2024.

[24] AlKhader W, Jayaraman R, Salah K, Sleptchenko A, Antony J, Omar M. *Leveraging blockchain and NFTs for quality 4.0 implementation in digital manufacturing.*, Journal of Manufacturing Technology Management. 24;34(7):1208-34; Oct 2023.

[25] Alam S, Bhatia S, Shuaib M, Khubrani MM, Alfayez F, Malibari AA, Ahmad S. *An overview of blockchain and IoT integration for secure and reliable health records monitoring.*, Sustainability. 23;15(7):5660; Mar 2023.

Edited by: Kavita Sharma
Special issue on: Recent Advance Secure Solutions for Network in Scalable Computing
Received: May 15, 2024
Accepted: Sep 1, 2024