# Decentralized Browser-based Cloud Storage: Leveraging IPFS for Enhanced Privacy

Georg Eilnberger
St. Pölten UAS
St. Pölten, Austria
e-mail: is211806@fhstp.ac.at

Timea Pahi
St. Pölten UAS
St. Pölten, Austria
e-mail: timea.pahi@fhstp.ac.at

Peter Kieseberg
St. Pölten UAS
St. Pölten, Austria
e-mail: lbkieseberg@fhstp.ac.at

***Abstract*—This work addresses the growing need for data management solutions that prioritize security, privacy, and user control, amidst the limitations of traditional centralized storage systems with a particular focus on the InterPlanetary File System (IPFS). The core objective is to explore the efficiency, challenges, and potential of IPFS in revolutionizing data storage and management. A significant contribution of this paper is the development of a proof-of-concept web application that employs IPFS for secure and efficient data handling. This application serves as a practical illustration of integrating IPFS into real-world data management scenarios. The security and performance of the application within the decentralized IPFS framework are thoroughly assessed. The study highlights the strengths of IPFS in ensuring data integrity and privacy while acknowledging the challenges in scalability and performance, particularly in handling large files and addressing WebRTC-TCP (Web Real-time Communication-TCP) socket incompatibility issues. Furthermore, we present recommendations for future enhancements of the proof-of-concept web application. These include improving direct file transfer capabilities, advancing file handling techniques, integrating robust key management solutions, and developing dynamic data replication strategies. The research in this paper underscores the potential of decentralized systems like IPFS in shaping the future of data storage, offering a more secure, private, and user-centric approach.**

***Keywords-IPFS; Data Privacy; Cloud Storage; Decentralized Storage.***

## I. INTRODUCTION

The field of data storage and access is experiencing a rise in popularity of decentralized models. Centralized data management systems, while established and efficient, present limitations in security, privacy, and user autonomy. This work examines the transition towards decentralized systems, with a focus on the IPFS and distributed data management principles. The motivation for this study arises from an increasing need for secure, private, and user-centric data management solutions. The research addresses several questions:

1) How do decentralized systems like IPFS compare with traditional centralized storage solutions in terms of efficiency, security, and data integrity?
2) What are the main challenges associated with the implementation and use of decentralized storage systems?
3) How can web applications effectively integrate decentralized systems like IPFS for data management, and what are the associated challenges and security implications?

The remainder of the paper is organized as follows. In Section II, we provide some required background and relevant related work, in Section III, we provide details on the design and implementation, whereas in Section IV the approach is evaluated with respect to security and performance. Section V provides some ideas for future work.

## II. BACKGROUND & RELATED WORK

### A. Decentralized File Systems

Decentralization in computing and in the web represents a shift from centralized to distributed control. This shift is not merely technical but also philosophical, highlighting ideas such as autonomy, resistance to censorship, and enhanced robustness against failures or attackers. In the early days of computing, centralized systems dominated due to their simplicity. However, the inherent drawbacks, such as single points of failure, scalability issues, and potential for abuse of power led to the exploration of decentralized alternatives [1].

The concept of decentralization in computing started taking shape with early developments. A pivotal development in this direction was the emergence of Peer-to-Peer (P2P) networks, characterized by their lack of reliance on central servers. Napster, one of the first widely used P2P networks, facilitated file sharing by allowing direct file transfers between users' computers. Despite its legal controversies, Napster demonstrated the potential for efficient, decentralized data distribution. Similarly, BitTorrent further advanced this model, efficiently handling large files and numerous simultaneous uploads and downloads, a significant step towards practical decentralized data sharing [2].

The advent of blockchain technology represented a critical development in decentralized systems. Blockchain's introduction of a popular tamper-proof ledger without central authority was first successfully implemented by Bitcoin, the initial decentralized digital currency. This implementation of blockchain technology demonstrated the feasibility of achieving consensus in a trustless environment Subsequently, the development of platforms such as Ethereum expanded the blockchain's applicability. Ethereum introduced functionalities like smart contracts, which allowed for a broader range of decentralized applications, illustrating the versatility and potential of blockchain technology in various domains [3].

### B. The InterPlanetary File System

IPFS is a protocol designed to create a peer-to-peer method of storing and sharing media in a distributed file system. Developed as a response to the limitations of the traditional centralized web storage model, IPFS represents a paradigm shift in how information is distributed and accessed [4].

A defining aspect of IPFS is its decentralized nature. Unlike conventional web storage solutions, which rely on centralized servers, IPFS distributes data across a network of nodes. This distribution of data not only mitigates risks associated with single points of failure but also enhances data accessibility and data permanence. Central to IPFS's functionality is content addressing. Traditional web uses location-based addressing, for example, URLs pointing to specific server addresses. In contrast, IPFS addresses content through its content itself by utilizing cryptographic hashing. This approach results in unique content identifiers (CIDs), making content retrieval more efficient and less redundant. Compared to location-based addressing, this method significantly improves both the efficiency and security of data storage and access. By relying on the content's cryptographic hash, immutability is inherent, allowing for verifiable data integrity and significantly contributing to the system's overall robustness [4].

Security in decentralized systems presents a unique set of challenges and considerations distinct from those in traditional centralized architectures. The decentralized nature, while offering advantages in terms of redundancy and resistance to certain types of attacks, also introduces specific vulnerabilities that must be addressed [5], [6].

*a) General Security Challenges in Decentralized Systems:* Decentralized systems face distinct security vulnerabilities. One key issue is the increased attack surface due to the distributed nature of these systems. Each node in a decentralized network can potentially become a target for attacks. Furthermore, in public decentralized systems, such as IPFS each participating node can potentially be malicious. To ensure that data remains unaltered and private over a distributed network robust encryption and validation mechanisms are required. However, implementing these effectively in a decentralized context, where control is inherently distributed, presents unique challenges in itself [7].

*b) Vulnerabilities in DHT-Based Routing Protocols:* Distributed Hash Table (DHT) based routing protocols, such as IPFS, have their own vulnerabilities. These include Sybil attacks, where an attacker tries to create a large number of fake identities/nodes to gain a disproportionately large influence on the network [8], [9], and Eclipse attacks, where the attacker isolates a single node or user from the rest of the network, potentially feeding it false and/or harmful information.

*c) Network Reliability:* Maintaining a consistent and reliable network is another critical challenge in decentralized systems. In IPFS, for instance, the absence or unavailability of nodes can lead to difficulties in data retrieval, highlighting the need for robust network health.

*d) Data Persistence and Redundancy:* In decentralized systems like IPFS, data persistence is dependent on nodes electing to store that data. Unlike centralized systems where data storage can be systematically managed and guaranteed, IPFS faces the challenge of ensuring that data remains available even when the node originally providing that data goes offline. This issue necessitates a redundant storage mechanism and an incentive for nodes to retain data.

## III. Approach

The web application developed as part of this work represents a proof-of-concept for a secure, decentralized file storage system. It operates within the broader ecosystem of IPFS, leveraging the decentralized nature of the platform to offer a novel approach to data storage and access. The application is hosted directly on IPFS, which provides a resilient and distributed hosting solution. This hosting choice aligns with the overarching theme of decentralization, ensuring that the application itself is as robust and distributed as the data it manages.

### A. Attacker models

In this work, we focus on the following three attacker models, as we consider them to be the most important ones with respect to IPFS:

1) Malicious IPFS Nodes: Given the open nature of IPFS, the application may interact with nodes that attempt to access or manipulate user data. To mitigate this, the application employs end-to-end encryption, ensuring that data remains secure and unreadable by unauthorized nodes.
2) Data Manipulation Attacks: The possibility of an attacker altering the data in transit is addressed through the use of IPFS's content addressing and the application's encryption mechanisms. The integrity of data is maintained as any alteration in the encrypted data will be detectable due to the change in its CID. Contrary to the next attack, this attacker might only try to redirect traffic or alter information, even without being able to actually decode it.
3) Eavesdropping Attacks: The risk of data interception is countered by encrypting the data before it is shared or stored on the network. This ensures that even if the data is intercepted, it remains incomprehensible to the attacker. Contrary to the previous attacker, this attacker is only passively involved, i.e. he/she does not change data.

The logic behind the selection of these three models is that one is an attack from inside the network, namely the most prominent one where a node is made malicious, while attacker model two and three model an active, as well as a passive, attacker respectively.

### B. Connectivity and File Processing Framework

In the current IPFS ecosystem, direct file transfers using the Helia library face challenges due to a WebRTC-TCP incompatibility issue in current IPFS nodes. As a pragmatic approach, the application utilizes third-party HTTP APIs for interactions with storage providers like Filebase or Pinata. This strategy is a temporary solution until direct transfer of files via IPFS becomes feasible. The use of HTTP APIs as a current means of file handling offers reliable storage and retrieval, albeit with a modest departure from the ideal decentralized model [10].

### C. Custom Identity Management and File Synchronization

This section elaborates on the unique approach to identity management and file synchronization within the developed web

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

application. The system hinges on the creation and utilization of a user-specific identity file, coupled with a dynamic file indexing mechanism, ensuring secure and efficient interactions with the IPFS network.

*a) Identity File Creation and Usage:* The first step is the creation of an identity file for the user. This pivotal process involves:

- Generation of an AES (Advanced Encryption Standard) private key, either supplied by the user or automatically generated by the application.
- Requirement for the user to input an API key from a chosen third-party storage provider. This is required due to current Helia limitations (WebRTC - TCP incompatibility).

The identity file, essentially a JSON object, encompasses crucial components for user identification and interaction with the IPFS network:

- The IPNS (InterPlanetary Name System) name, pointing to the file index JSON object
- The user's AES private key
- The user's third-party API key

This identity file represents the core of user data portability, enabling access to their IPFS storage from any device by merely transferring this file.

*b) File Structure and Index File Mechanism:* Every file, including the index file, adheres to a structured format:

- Composed as JSON objects
- Contains encrypted data as a Uint8-Array
- Includes the AES-GCM initialization vector it was encrypted with

Upon uploading the first file, the index file is generated, and an IPNS entry is created to consistently point to the latest CID of this index file. The index file plays a crucial role in the system:

- Structured as a JSON object.
- Contains an encrypted list of all files, each entry detailing:
  - File CID
  - File name
  - File size
  - SHA-256 hash of the file
  - Optional metadata for enhanced file information (e.g. a timestamp of the last change).

Currently, this approach does not account for collisions, as 256 bit hashes have a wide result space, thus the probability of accidental collisions is very low. Still, this could be improved in future versions.

*c) File Retrieval Process:* The steps to retrieve a file are as follows:

1) Connect to IPFS and query the CID of the index file JSON object.
2) Download the index file JSON object.
3) Decrypt the file list using the attached AES-GCM initialization vector and the user's AES256 private key.
4) Display metadata of all files contained in the index.
5) On file request, query the specific CID.
6) Download the requested file JSON object.
7) Decrypt the file using the attached AES-GCM initialization vector and the user's AES256 private key.

*d) File Storage Process:* When a new file is uploaded or an existing one modified, the following steps are undertaken:

1) Encrypt the file using the user's AES256 private key and a newly generated initialization vector.
2) Create a JSON object for the file, storing the encrypted data and the initialization vector.
3) Upload the file to an IPFS storage provider (using Helia or third-party HTTP APIs).
4) Record the file's CID and metadata, appending it to the index file.
5) Request the storage provider to pin the new file on IPFS.
6) Upon successful pinning, upload the updated index file and request pinning.
7) Update the IPNS entry to reflect the new index file.
8) Unpin the old index file (and the old file if it was an update) after successful IPNS update and new file pinning.

This architecture ensures a secure, user-friendly, and efficient mechanism for managing files on the decentralized IPFS network, addressing current limitations while laying the groundwork for future improvements in direct file transfer capabilities.

### D. Encryption in the Application

The approach implements AES-GCM, an Advanced Encryption Standard in Galois/Counter Mode, for its data encryption and decryption processes. AES-GCM is chosen primarily due to its integration with the Web Crypto API, along with its Authenticated Encryption with Associated Data (AEAD) properties and widespread hardware acceleration support. The selection of AES-GCM for encryption in the application is driven by several factors:

- Web Crypto API Compatibility: AES-GCM is readily available in theWeb Crypto API, facilitating easy implementation in web applications [11].
- AEAD: AES-GCM provides both encryption and data integrity, ensuring data confidentiality and protection against tampering [12].
- Hardware Acceleration: The widespread hardware support for AES that allows for fast computation with cheap hardware.

## IV. Evaluation and Conclusions

### A. Security evaluation

The web application was designed with specific attacker models in mind, primarily focusing on safeguarding user data from unauthorized access and manipulation. With respect to the attacker models outlined in Section III, the following conclusions could be drawn:

1) Malicious IPFS Nodes: The primary threat comes from malicious nodes within the IPFS network that may attempt to access or tamper with user data. The application's use of AES-GCM encryption effectively counters this threat by ensuring data confidentiality. Encrypted files, even if intercepted, remain inaccessible to unauthorized parties.

2) Data Manipulation Attacks: Another concern is the potential for data manipulation during transmission. The self-verifying nature of IPFS CIDs, combined with the integrity assurance of AES-GCM, provides robust protection against such attacks. This dual layer of security ensures that any tampered data is easily detectable.
3) Sybil and Eclipse Attacks: While the application does not directly mitigate DHT vulnerabilities like Sybil and Eclipse attacks, it minimizes their impact on user data privacy. The encrypted data stored on IPFS remains secure against these attacks, as the encryption layer acts independently of the underlying DHT's vulnerabilities [7], [9].

In addition, the use of AES-GCM for encryption plays a crucial role in securing user data:

- Data Confidentiality: AES-GCM ensures that file contents remain confidential. By encrypting data before it is uploaded to IPFS, the application prevents unauthorized access, even if the data is replicated across potentially untrustworthy nodes.
- Data Integrity: Alongside confidentiality, AES-GCM provides data integrity checks. This feature is critical in a decentralized setting where data passes through multiple nodes, as it enables the detection and rejection of tampered data.
- Performance Considerations: While AES-GCM is computationally efficient due to widespread hardware acceleration support, the application's encryption process is dependent on the user's device capabilities. This can impact performance, particularly for larger files.

In conclusion, the security evaluation reveals that the web application effectively addresses key security concerns within the decentralized IPFS framework. The robust encryption strategy ensures data confidentiality and integrity, mitigating risks associated with decentralized data storage and transmission. The application's current security measures provide a solid foundation, though future enhancements could focus on advanced key management and addressing broader DHT vulnerabilities.

### B. Performance and Stability

This section focuses particularly on efficiency in handling large files, WebRTC and TCP socket compatibility issues, and the implications of these factors on data loss prevention and redundancy strategies.

*1) Efficiency and Large File Handling:* The application's current architecdture faces challenges in managing large files due to limitations in the splitting and handling of large data sets. Key observations include that large files lead to extended encryption and upload times, constrained by the device's RAM and processing power. Furthermore, there are also some concerns regarding scalability: Without the ability to split large files into manageable blocks, the application's scalability is hindered, particularly when dealing with extensive data sets or high-volume storage requirements.

*2) WebRTC and TCP Socket Incompatibility:* One of the primary limitations in the current implementation of the application is the incompatibility between WebRTC and TCP sockets within the IPFS ecosystem. This limitation impacts the stability of the application. Regarding connection limitations, due to this incompatibility, the application primarily relies on a few nodes that act as gateways for browser-based interactions. This reliance can lead to bottlenecks and potential points of failure [10]. Furthermore, the reliance on HTTP APIs could be a problem: The application currently uses HTTP APIs of third-party IPFS storage providers like Filebase or Pinata for file handling, which, while reliable, deviates from the ideal decentralized model and could impact long-term scalability and decentralization goals [10].

*3) Data Loss Prevention and Redundancy:* In addressing the concerns of data loss and ensuring redundancy, the application leverages the inherent strengths of the IPFS network. Regarding the decentralized storage, data availability in IPFS is independent of the storage location, allowing for convenient replication and enhanced scalability. Utilizing crypto-based storage providers like Filecoin offers a cost effective solution for redundant storage. At the time of writing, the cost of storing 10TB of data on Filecoin (1.95 USD per month) is significantly lower than traditional cloud storage options like Amazon S3 (235 USD per month) [13], [14]. Regarding collateral-based reliability, Filecoin storage providers are required to provide collateral, adding an additional layer of reliability and commitment to data preservation [15].

In conclusion, the evaluation of the application's performance and stability highlights key areas for improvement, particularly in large file handling and direct file transfer capabilities. Despite these challenges, the application benefits from the decentralized, scalable nature of IPFS and the cost-effective, redundant storage solutions offered by crypto-based storage providers. Future work should focus on enhancing file transfer capabilities and exploring more efficient file processing methods to bolster the application's performance and scalability within the decentralized web.

## V. Future Work

Based on the findings, several recommendations are proposed to advance the field in terms of future work: Regarding technical improvements, a lot of improvements could be done by addressing the limitations identified in IPFS, particularly in file transfer and with respect to compatibility. Regarding key management, fruitful future work lies in investigating more sophisticated encryption key management techniques, including hardware solutions like TPM2 [16], which could significantly improve security. Regarding automated data replication, developing mechanisms for user-defined data replication will increase redundancy and reliability, especially using cost-effective storage solutions like Filecoin.

## References

[1] M.-Å. Hugoson, "Centralized versus decentralized information systems: A historical flashback", in *History of Nordic Computing 2: Second IFIP WG 9.7 Conference, HiNC2, Turku, Finland, August 21-23, 2007, Revised Selected Papers 2*, Springer, 2009, pp. 106–115.

[2] P. Raj *et al.*, "High-performance peer-to-peer systems", *High-Performance Big-Data Analytics: Computing Systems and Approaches*, pp. 317–337, 2015.

[3] W. Wang *et al.*, “A survey on consensus mechanisms and mining strategy management in blockchain networks”, *Ieee Access*, vol. 7, pp. 22 328–22 370, 2019.

[4] IPFS, “Official ipfs documentation”, retrieved: March 2025, [Online]. Available: https://docs.ipfs.tech/.

[5] E. Karaarslan and E. Konacaklı, “Data storage in the decentralized world: Blockchain and derivatives”, *arXiv preprint arXiv:2012.10253*, 2020.

[6] N. Z. Aitzhan and D. Svetinovic, “Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams”, *IEEE transactions on dependable and secure computing*, vol. 15, no. 5, pp. 840–852, 2016.

[7] B. Prünster, A. Marsalek, and T. Zefferer, “Total eclipse of the heart–disrupting the {interplanetary} file system”, in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3735–3752.

[8] L. Wang and J. Kangasharju, “Real-world sybil attacks in bittorrent mainline dht”, in *2012 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2012, pp. 826–832.

[9] J. R. Douceur, “The sybil attack”, in *International workshop on peer-to-peer systems*, Springer, 2002, pp. 251–260.

[10] ipfs-helia, “Helia github repository.”, retrieved: March 2025, [Online]. Available: https://github.com/ipfs/helia/issues/256.

[11] Mozilla Foundation, “Web crypto api”, retrieved: March 2025, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API.

[12] D. McGrew, “An interface and algorithms for authenticated encryption”, Tech. Rep., 2008.

[13] Amazon Web Services, “Amazon s3 pricing”, retrieved: April 2024, [Online]. Available: https://aws.amazon.com/s3/pricing/.

[14] Storage.market, “Ipfs storage market”, retrieved: April 2024, [Online]. Available: https://file.app/.

[15] Filecoin, “Official filecoin documentation”, retrieved: March 2025, [Online]. Available: https://docs.filecoin.io/.

[16] K. Shang *et al.*, “Cluster nodes integrity attestation and monitoring scheme for confidential computing platform”, in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, IEEE, 2023, pp. 740–749.