Article

Efficient and Secure Distributed Data Storage and Retrieval Using Interplanetary File System and Blockchain

Muhammad Bin Saif [†], **Sara Migliorini** [*,†] and **Fausto Spoto** [†]

Department of Computer Science, University of Verona, 37134 Verona, Italy; muhammadbin.saif@univr.it (M.B.S.); fausto.spoto@univr.it (F.S.)
* Correspondence: sara.migliorini@univr.it
† These authors contributed equally to this work.

Abstract: Blockchain technology has been successfully applied in recent years to promote the immutability, traceability, and authenticity of previously collected and stored data. However, the amount of data stored in the blockchain is usually limited for economic and technological issues. Namely, the blockchain usually stores only a fingerprint of data, such as the hash of data, while full, raw information is stored off-chain. This is generally enough to guarantee immutability and traceability, but misses to support another important property, that is, data availability. This is particularly true when a traditional, centralized database is chosen for off-chain storage. For this reason, many proposals try to properly combine blockchain with decentralized IPFS storage. However, the storage of data on IPFS could pose some privacy problems. This paper proposes a solution that properly combines blockchain, IPFS, and encryption techniques to guarantee immutability, traceability, availability, and data privacy.

Keywords: blockchain; decentralized storage; IPFS; data traceability

check for updates

Citation: Bin Saif, M.; Migliorini, S.; Spoto, F. Efficient and Secure Distributed Data Storage and Retrieval Using Interplanetary File System and Blockchain. *Future Internet* **2024**, *16*, 98. https://doi.org/10.3390/fi16030098

Academic Editor: Qiang Qu

Received: 27 February 2024
Revised: 11 March 2024
Accepted: 13 March 2024
Published: 15 March 2024

1. Introduction

In the digital information age, the exponential growth in data volume has posed significant challenges in terms of efficiency and scalability. Indeed, traditional centralized data storage and retrieval systems act as a single computational node, which consequently become a trust bottleneck and a single point of failure. To overcome these challenges, data storage is shifting to distributed systems, where data spread across multiple public nodes. In distributed systems, the decentralized peer-to-peer nodes can synchronize with each other without the need for a central authority. For instance, a torrent file-sharing protocol [1] uses a peer-to-peer network to connect all participating computers, allowing them to share files in a decentralized manner.

More recently, blockchain technology has emerged as an innovative solution for reaching a consensus about a global state in a decentralized manner, without the need for a central authority. It allows a decentralized network of untrustworthy nodes to agree on the content and state of the blockchain, independently from each other. Furthermore, the information stored in blockchain is also immutable, since the content of a block cannot be modified without invalidating the content of all the subsequent blocks. In blockchain, smart contracts play a vital role in developing decentralized applications on top of the blockchain. In this context, a smart contract is a piece of code deployed in blockchain to execute predefined actions, enabling automated execution by ensuring the immutability and transparency of agreements in a trustless environment [2]. Therefore, smart contracts can significantly contribute to the spread of blockchain technology in several application domains, such as healthcare [3], tourism [4], energy and water management [5], identity management [6], supply chains [7], and so on.

In contrast to traditional centralized solutions that lack data traceability and transparency, blockchain and smart contract technology could help to overcome these challenges.

Indeed, blockchain technology inherently provides traceability, immutability, and trustworthiness [8]. These properties can secure and tamper-proof critical information in many domains. The final aim is to increase the trust of end users by issuing non-repudiable certificates. On the other hand, storing large amounts of data in blockchain is not feasible, due to scalability and transaction cost challenges. Indeed, increasing data volume increases the transaction cost and decreases the blockchain throughput (scalability). Therefore, as the volume of data grows, blockchain solutions become increasingly inefficient and economically unviable. However, a solution, called off-chain data storage [9], has been proposed to address the challenges of efficient data storage strategies, minimizing on-chain data while ensuring integrity. The idea is to develop proper solutions that guarantee the consistency and immutability of off-chain data from those of the on-chain data.

The Interplanetary File System (IPFS) is a peer-to-peer distributed system for storing, accessing, and sharing files, websites, applications, and data. First introduced in 2015, IPFS developed upon a decentralized environment and incorporates distributed and bandwidth-saving techniques from torrent [10]. Blockchain and IPFS function as decentralized technologies but serve different purposes and have distinctive characteristics. IPFS offers an efficient, peer-to-peer decentralized public network for large distributed data storage and access. It aims to improve the efficiency and resilience of traditional web protocols by allowing files to be stored in multiple locations, making them resistant to censorship and ensuring availability, even if some nodes go offline. On the other hand, blockchain serves primarily as a decentralized ledger, recording transactions or data transparently and in a tamper-proof way. Integrating these technologies represents an efficient solution to the challenges mentioned above related to scalability, efficiency, immutability, and data availability.

One of the main characteristics of IPFS is that anyone can access the data stored on this public network. Therefore, this can limit its usability if the stored data present privacy concerns. In the domain of digital data security, hashing and encryption are considered sophisticated cryptographic practices. Hashing, the process of generating a unique fixed-size hash value from any input data, of any size, serves as a critical tool in ensuring *data integrity*. This procedure is deterministic, providing that any change in the data, no matter how small, results in a significantly modified hash value, allowing the effective detection of data tampering or corruption. Encryption, on the other hand, converts original data, known as plaintext, into an encoded version known as ciphertext. This transformation, controlled by sophisticated algorithms and cryptographic keys, ensures that unauthorized entities cannot access the data, providing *data confidentiality*. In distributed data storage systems, combining hashing and encryption offers a robust framework for data security, effectively tackling two critical aspects of data security: integrity and confidentiality. Indeed, hashing values can be used to provide a unique fingerprint for the data that allows one to check data integrity without the need to decrypt data and compare the original contents. Conversely, the sole use of encryption techniques requires the decryption of the retrieved data, namely, the inverse transformation from ciphertext to plaintext, exposing private keys and posing a significant security risk.

The solution proposed in this paper integrates IPFS with blockchain technology to solve all the problems above. In particular, we propose to exploit the decentralized and efficient architecture of IPFS for large data storage, significantly reducing costs while improving scalability. We propose a dual-layer security mechanism that combines hashing and encryption to ensure robust data security and privacy. We propose a novel approach for verifying data integrity, which detects and handles any modifications to the data by generating and comparing hashes of the retrieved and stored information. Furthermore, we eliminate the need for decryption during data retrieval and querying, thus mitigating the risks of exposing private keys and improving overall system security. The proposed approach is developed with reference to a real-world scenario in which a set of IoT devices periodically produces several information records that need to be stored in an immutable and tamper-proof way, as well as preserving their confidentiality.

The remainder of the paper is organized as follows: Section 2 summarizes some previous results in combining IPFS, blockchain, and encryption techniques for distributed data storage and security. Section 3 formalizes the considered problem, while Section 4 illustrates the proposed solution. Section 5 discusses the implementation of the solution and the performed experiments. Finally, Section 6 concludes the work.

2. Related Work

The integration of IPFS and blockchain technology offers a significant advancement in distributed data storage and security. This section reviews existing research on blockchain-based data storage, using IPFS, encryption, and hashing techniques to ensure data security and privacy.

2.1. Blockchain and IPFS for Data Storage

Blockchain technology is receiving a lot of attention due to its potential for secure and decentralized data storage. The studies on blockchain mainly explored the blockchain's immutable ledger system for data storage; however, they frequently emphasized the limits associated with the high costs and inefficiencies of keeping large volumes of raw data directly in the blockchain. In [11], the authors propose an in-depth analysis of these challenges, emphasizing the need for more effective data storage solutions within the blockchain framework. In this regard, IPFS is becoming increasingly common as a large data storage solution, thanks to its decentralized and effective design. In [12], the authors employ blockchain technology to transmit data in a peer-to-peer network and IPFS as data-sharing infrastructure to share pre-trained deep learning models to stakeholders. Meanwhile, in [13], the authors propose a blockchain and cloud-based decentralized secure storage for data availability, privacy, and efficient resource utilization. The work in [14] proposes the integration of blockchain and IPFS, showing the potential for secure data retrieval and storage. The proposed approach leverages different privacy modes for data privacy and security by storing nonsensitive data on IPFS without encryption and sensitive data with two-layered encryption. This approach mitigates the risk of a single point of failure and data tampering. However, data retrieval and query require the decryption of data stored on IPFS, which may introduce additional security and privacy risks associated with the exposure of private keys.

2.2. Data Security and Privacy Methods

Data privacy has been a significant challenge in decentralized systems. Encryption and hashing have been pivotal in securing data within the blockchain and IPFS frameworks. The literature on various encryption algorithms and hashing techniques studied by [15] shows the advancement in these methods in maintaining data integrity and security. Furthermore, ref. [16] focuses on the role of these techniques in preserving data privacy in decentralized systems and on the importance of a balance between accessibility and security. In [17], the authors delve into privacy-preserving approaches such as zero-knowledge proofs and homomorphic encryption. These studies emphasize the development of techniques that secure data without compromising privacy. However, these techniques can provide solutions for data security and privacy but often fall short in terms of balancing data security with system efficiency, such as computation overhead, scalability, and dynamic data handling. In addition, relying on encryption inadvertently exposes private keys during decryption processes, making it a substantial security risk. The work in [18] proposes a blockchain-based framework for privacy preservation and secure access control in cloud storage. The proposed scheme combines the Ethereum blockchain and ciphertext-policy attribute-based encryption (CP-ABE) for user data security and privacy. However, due to privacy issues and data leakage, the cloud might not be a trustworthy source for data storage.

2.3. Query Optimization Techniques

Leveraging blockchain and IPFS in decentralized storage systems has transformed data storage and retrieval, particularly in sectors demanding high integrity and efficiency. These studies illustrate how blockchain and IPFS can improve query optimization, security, and scalability in distributed storage systems. In [19], the authors focus on agricultural product traceability, using IPFS to store multiple data types and blockchain for securing IPFS hash addresses to improve query efficiency and data authenticity. However, the sensor data are collected and processed on a centralized private server before storing data on IPFS. Therefore, this approach may pose security risks related to a single point of failure, data loss, or compromised server. In [20], the authors integrate blockchain, IPFS, and Elasticsearch to address big data storage challenges in distributed systems, resulting in reduced storage overhead, search latency, and access control. The proposed approach relies on blockchain's inherent security features. Moreover, the elastic search approach focuses primarily on search latency and precision, without considering the security implications of query handling mechanisms. The work in [21] proposes a secure and decentralized framework for managing cloud data provenance by integrating blockchain technology with the IPFS. The proposed solution ensures provenance data availability, security, and integrity by utilizing decentralized storage and blockchain for immutability. The method for verifying data integrity is secure; however, it might be computationally intensive, particularly in scenarios where the frequent validation of large data sets is necessary.

Despite these advancements, existing solutions have faced challenges, particularly in their reliance on decryption, which has associated security risks. Namely, the exposure of private keys remains a major concern, as discussed in [22]. This justifies the importance of a solution that could ensure data security and privacy without relying on decryption. The solution proposed in this paper addresses these challenges by optimizing and securing data storage on IPFS without relying on decryption. This novel approach, based on storing only the Content Identifier (CID) in blockchain and the actual data on IPFS, along with sophisticated encryption and hashing techniques, is a significant step forward from traditional methods. This innovation enhances data privacy and improves query optimization, as detailed in our methodology.

3. Problem Statement and Formalization

In a typical IoT scenario, several IoT devices or sensors produce a set of data every day that needs to be stored and certified to ensure immutability, traceability, and tamper-proofing. In this scenario, it is crucial for the produced data to be stored permanently without any subsequent modification or deletion.

Given such premises, we can formalize the following properties that one wants to ensure by using blockchain and smart contract technology in conjunction with the IPFS protocol.

Property 1 (data immutability)**.** *Given a piece of data d stored in a database or file system, we say that d is immutable if it cannot be modified after its storage, or otherwise, any subsequent modification of d can be easily identified.*

Blockchain technology ensures the immutability of data stored inside its blocks. Indeed, block confirmation ensures that there are no further data modifications to the block. Typically, the amount of data stored in blockchain is minimal due to technological challenges and cost constraints. However, blockchain can ensure the immutability of data stored off-chain by essentially storing a fingerprint (or hash) of the data. Such a fingerprint is enough to identify if some information stored outside the blockchain has been modified.

In many real-world scenarios, immutability can be considered too strict since data could need to be updated or rectified due to some missing or erroneous parts. Therefore, we consider immutability together with another property called traceability.

Property 2 (data traceability)**.** *Data traceability refers to the ability to follow the data transformations back to their origin, verifying the authenticity and integrity of the data. It can also be intended as the degree to which a system or a data provider can record the changes made to the data.*

Combined with immutability, traceability can lead to a more robust system where each stage of the data life can be stored immutably. Instead of preventing the possibility of modifying data, these two properties together allow the registration of several subsequent evolutions of the data, like in a versioning system. Blockchain has emerged as a possible solution to implement data traceability by creating an information trail that ensures security and data availability [7].

The last requirement is related to the system used to store data off-chain.

Property 3 (data availability)**.** *Data availability refers to a user's confidence that the stored data and all the information required to verify specific properties of that (such as immutability) are available in a given period.*

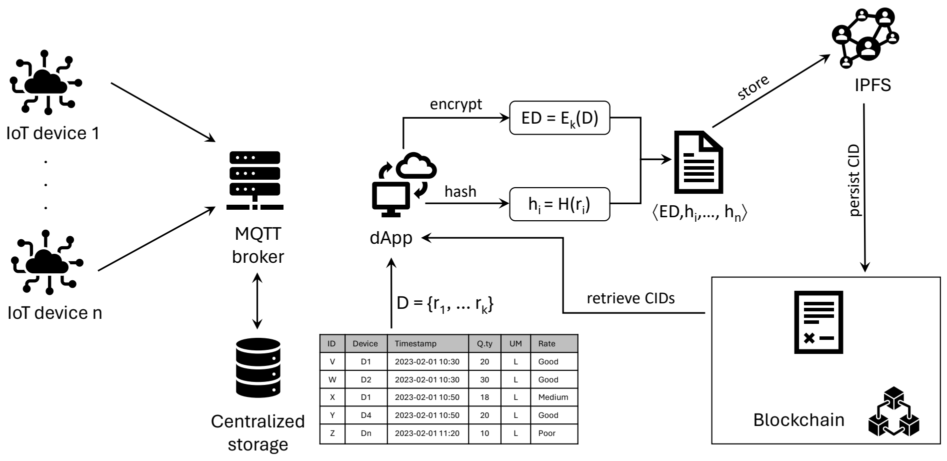
Given such desired properties, we can contextualize the considered problem as follows: Suppose we have a set of IoT devices that continuously produce a set of measurements of a predefined quantity, for instance, the amount of water that flows through a pump, or the temperature in a greenhouse, or the quantity of precipitation. These pieces of data are collected through a Message Queuing Telemetry Transport (MQTT) broker, which stores the received data in centralized storage, such as a relational database, a set of log files, and so on. In this scenario, the problem we want to solve is making such information immutable and tamper-proof, providing the traceability properties mentioned above and ensuring their availability.

The following section illustrates the proposed solution, which integrates blockchain, smart contract technology, and the IPFS decentralized storage.

4. Proposed Solution

Figure 1 illustrates the general architecture of the proposed solution. It includes at its core a decentralized application (dApp), which interacts with the traditional centralized storage for retrieving the data to be persisted, made immutable through the blockchain by using a set of smart contracts, and the IPFS. It is important to note that the main idea is to store data efficiently on IPFS rather than directly in blockchain, which can be costly and inefficient. However, since data are supposed to be sensitive and subject to privacy concerns, it undergoes a process of encryption before being stored on IPFS. At the same time, the data hashing technique generates a fingerprint of each row and allows the fast detection of data manipulation without the need to decrypt the data. Moreover, the IPFS returns a CID as the result of storing data, and the CID is permanently stored in blockchain for traceability. The CID is a unique identifier for the data and essentially a hash of the data stored on IPFS. While in a centralized web service, content or data can be accessed through location-based uniform resource locators (URLs); in IPFS, content or data are stored and retrieved based on hash, which makes it more challenging to censor or manipulate. Moreover, there is no need for trust in IPFS data hosting entities.

The proposed solution achieves data privacy, integrity, and query optimization by properly integrating the hashing and encryption of data. In the future, any modification can be detected by simply comparing the previous hash stored on IPFS and newly generated hashes of modified data. Consequently, this approach eliminates the need for decryption, which often depends on revealing private keys and poses security risks, limiting this possibility only to the case in which previous data need to be effectively restored. Moreover, this scheme allows for the easy separation of concerns since an integrity check can be performed without revealing the content.



ID	Device	Timestamp	Q.ty	UM	Rate
V	D1	2023-02-01 10:30	20	L	Good
W	D2	2023-02-01 10:30	30	L	Good
X	D1	2023-02-01 10:50	18	L	Medium
Y	D4	2023-02-01 10:50	20	L	Good
Z	Dn	2023-02-01 11:20	10	L	Poor

Figure 1. The architecture of the proposed solution.

Algorithm 1 illustrates how the data from the centralized storage are prepared for storage in IPFS. The function CREATEDATAFORIPFS receives as input a list of records corresponding to the database rows in Figure 1. For each row, the function computes the hash of the row content (see line 4) and includes the tuples $\langle id_i, H(record_i) \rangle$ in a list, where $H()$ represents the application of the hashing function. After that, the function encrypts the overall content through the function $E()$ (see line 6). The content stored on the IPFS is the union of these two elements, as reported in line 7. We employ the SHA-56 algorithm [23] for hashing, which maps the given data to a 256-bit fixed-size cryptographic hash. The characteristic of a hash is that all unique input data produce a unique hash value. Even minor changes in the input data result in a significant change in the hash output. This property of hashing is known as the avalanche effect.

Algorithm 1 Creation of the data to be stored in IPFS

```
1: procedure CREATEDATAFORIPFS($rawData = \{\langle id_i, record_i \rangle\}_{i=1}^{n}$)
2:     $hashedData = \{\}$
3:     for $d \in rawData$ do
4:         $hashedData = hashedData \cup \{\langle id_i, H(record_i) \rangle\}$
5:     end for
6:     $encryptedData = E(rawData)$
7:     return $combinedData = \{hashedData, encryptedData\}$
8: end procedure
```

Definition 1 (Hashing). *Let D be the data entry and H be the hash function implementing the SHA-256 algorithm. The hashing process can be represented as*

$$H(D) = h \tag{1}$$

where h is the unique hash result.

For encryption, we employ Password-Based Key Derivation Function 2 (PBKDF2) [24], a key derivation function with a sliding computational cost to reduce vulnerability to brute-force attacks. It generates a cryptographic key from a secret, preventing the need to save the encryption secret on a centralized database or server and improving security.

Definition 2 (Key Derivation). *Let P represent the user-provided secret and KDF the key derivation function. The key K used for the encryption and decryption process can be derived as follows:*

$$K = KDF(P, s, c, dkLen) \tag{2}$$

where s is the salt, c is the number of iterations, and dKLen is the desired key length.

One of the novelties of the proposed solution resides in the fact that it does not store the encryption secret or keys. Instead, it encrypts a known string S with the secret and stores this encrypted string on IPFS. Algorithm 2 illustrates the process of verifying a secret without storing the secret or keys anywhere. Given the secret P' provided by the user, the corresponding key K' is derived using the KDF function. Then, the encrypted string $E(S)$ retrieved from IPFS is decrypted by using K'. If the obtained decrypted string S' corresponds to the original string S, then the provided secret is correct, and the verification is successful; otherwise, the verification fails.

Algorithm 2 Verification of a secret S without revealing it

```
1:  procedure VerifySecret($P'$)
2:      Retrieve the encrypted string $E(S)$ from IPFS.
3:      $K' = KDF(P', s, c, dkLen)$
4:      $S' \leftarrow D(E(S), K')$
5:      if S′ = S then
6:          the secret is correct
7:      else
8:          the secret is incorrect
9:      end if
10: end procedure
```

If the secret verification is successful, the data are encrypted using the derived key K', as reported in line 6 of Algorithm 1. The encrypted data are then combined with the hashed data inside a JSON object (see line 7 of Algorithm 1).

Definition 3 (Encryption)**.** *Let E represent the encryption function, K the key derived from the secret, and D the data. The encrypted data ED can be obtained as follows:*

$$ED = E(K, D) \tag{3}$$

Once the *combinedData* are generated, it is stored on IPFS. IPFS generates a unique CID, representing a reference for the encrypted and hashed data on IPFS. The blockchain is the final step of the proposed architecture. After storing the encrypted data and the hashed data on IPFS and obtaining the corresponding CID, this CID is stored on the blockchain through a smart contract. Such a smart contract will maintain a list of CIDs related to the data stored on IPFS. The immutability of blockchain ensures that the CID is recorded and cannot be changed or removed, which provides an auditable track of the data integrity and traceability. The primary purpose of this smart contract is not only to store CIDs permanently but also to retrieve and verify the persisted data.

Algorithm 3 illustrates the function DataVerification, which performs an integrity check about the current content of the centralized database and what has already been stored in the IPFS. After retrieving both contents, D_{api} and D_b, respectively, the function analyzes if each row in the database is already contained in the blockchain and, in this case, if the hashes are the same. The newly identified records are then stored in IPFS, as illustrated in Algorithm 1, and a transaction gets performed to store the corresponding new CID. Conversely, the records whose hashes do not coincide become classified as corrupted or inconsistent. This consistency verification does not need to decrypt the stored data but is based only on comparing the hashes. However, based on the specific policies of the system, in case of corrupted or inconsistent data, their original version could be eventually retrieved through a decryption of the encrypted data contained at the corresponding CID.

The proposed solution allows for continuous monitoring of data updates in an efficient and secure way. We improved the system's privacy and security by not decrypting the sensitive data but only comparing hashes. Furthermore, by employing blockchain as an immutable storage for comparisons, we ensure that the verification process is tamper-proof and auditable. The robust integration of cryptographic hashing and encryption, IPFS, and blockchain enables a sophisticated solution that ensures data integrity, enhances transparency, and maintains confidentiality in a public network.

Algorithm 3 Data retrieval and verification

```
 1: procedure DATAVERIFICATION
 2:     $D_b \leftarrow$ retrieve from IPFS the documents with the stored CIDs
 3:     $M_b \leftarrow$ = map from the hashed data in $D_b$ where key = $id_i$ and value = $H(r_i)$
 4:     $D_{api} \leftarrow$ retrieve from the API the data stored in the database
 5:     for $r \in D_{api}$ do
 6:         $id \rightarrow$ identifier of the current record $r$
 7:         $h_{new} \rightarrow H(r)$
 8:         $h = M_b.get(id)$
 9:         if $h$ is null then
10:             the data entry is new and needs to be persisted in the blockchain
11:         end if
12:         if $h == h_{new}$ then
13:             the data entry has not been modified
14:         else
15:             the data entry has been modified
16:         end if
17:     end for
18: end procedure
```

5. Evaluation

This section presents the proposed solution implementation and mainly focuses on testing the system performance and security with respect to different data sizes and security vulnerabilities.

5.1. Experimental Setup

The experiments have been performed on a computer with an Intel Core i5 CPU of 1.00 GHz and 8 GB of RAM. A dApp has been developed in React (https://react.dev/, accessed on 12 March 2024) using the TypeScript language (https://www.typescriptlang.org/, accessed on 12 March 2024), CryptoJS (version 4.2.0) for cryptographic functionalities, and the ipfs-http-client (version 60.0.1) for IPFS decentralized storage interactions. Additionally, we employ the AES algorithm for encryption and decryption with a key length of 256 bits derived from the secret. To ensure the integrity of the data, we leverage the SHA256 hashing algorithm. The proposed solution uses IPFS for data storage and retrieval, with Infura as the IPFS node provider and Ethereum blockchain for storing the IPFS CID to ensure robust and decentralized storage. Table 1 presents a detailed description of the experimental setup. The source code of the solutions has been made available through a GitHub repository (https://github.com/MuhammadBinSaif/Efficient-and-Secure-Distributed-Data-Storage-and-Retrieval-Using-IPFS-and-Blockchain, accessed on 12 March 2024).

Table 1. Experimental Setup Specifications.

Specification	Details
Front-end Framework	React 18.2.0
Programming Language	TypeScript
Encryption Library	CryptoJS 4.2.0
IPFS Client	ipfs-http-client 60.0.1
Encryption Algorithm	AES
Key Generation Method	PBKDF2
Hashing Algorithm	SHA256
IPFS Node Provider	Infura
Blockchain	Ethereum
Processor	Intel Core i5 CPU 1.00 GHz
RAM	8.00 GB

5.2. Performance Evaluation

This section presents a performance evaluation of the proposed system. We adopted two different metrics to measure that performance: the throughput and the total time. The aim is to show the impact of changing data size on the system throughput and on the time taken to encrypt and decrypt the JSON data. We generated dummy JSON data (that resemble real-life JSON data) by using the Faker library (https://fakerjs.dev/, accessed on 12 March 2024).

5.2.1. Encryption Throughput

The throughput T is measured in terms of the amount of data D processed per unit time t during the encryption and the decryption phases. In particular, given the size of the encrypted data $size(ED)$ and of the decrypted data $size(DD)$, the relative throughput can be computed as follows, respectively:

$$T_E = \frac{size(ED)}{t} \qquad\qquad T_D = \frac{size(DD)}{t} \tag{4}$$

We evaluated the throughput of the proposed solution on a wide range of data sizes from 10 kb to 3000 kb. Figure 2 illustrates that the system's throughput increases as data size increases. Initially, the throughput was around 3.07 kilobits per second (kb/s) for 10 kilobits (kb) of data. However, with larger data sizes, such as 3000 kb, the throughput has increased to over 307.12 kb/s. The throughput decreases for smaller amounts of data because of the significant time taken by the initial encryption or decryption steps and inherent computational efficiencies at larger data scales. However, as the data volume increases, these initial steps become less significant in the overall process. These results demonstrate that AES is scalable and efficient in processing large data sets and suitable for applications dealing with large data volumes, resulting in secure and efficient data encryption and decryption.

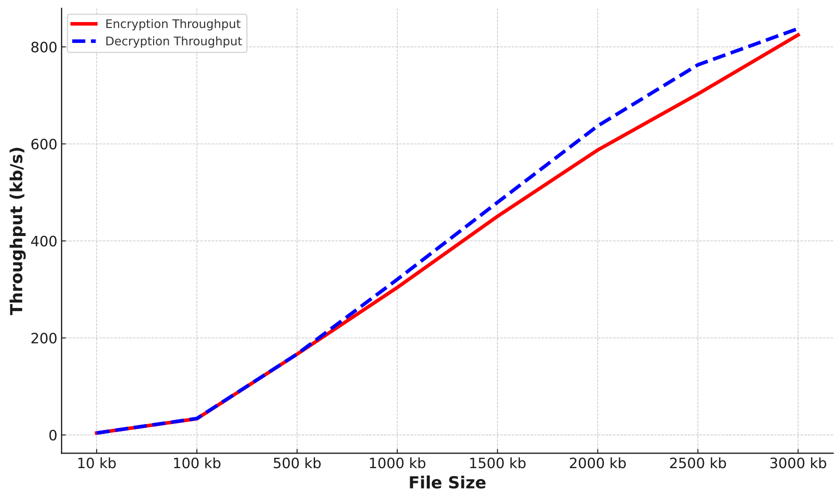


Figure 2. Throughput in kb/s.

5.2.2. Total Time

The total time $t = t_e - t_i$ is the difference between the initial time t_i when a process starts and the end time t_e when a process is completed. This section evaluates the total time of the proposed solution for the encryption, decryption, and generation of the hash, and for storing the hashed and encrypted data on IPFS for different data sizes, ranging from 10 kb to 3000 kb. Figure 3 illustrates the total time for these processes. The results show that the time for each operation increases as data size increases. The processing time for encryption, decryption, and hashing of 10 kb of data is approximately 2472, 2511, and 1 ms, respectively. In contrast, storing 10 kb of data on IPFS took 720 ms. As data size increases to 3000 kb, we can observe an increase in operation times for encryption to 3647 ms, decryption to 3590 ms, hashing to 210 ms, and storing on IPFS to 6252 ms. The gradual increase in processing time with data size demonstrates the efficiency of the AES and SHA256 algorithms. However, because of network latency, data propagation, and processing delay, IPFS requires a longer time to store data. However, this time could also be considered acceptable for real-world applications since this immutable persisting activity will be reasonably conducted offline, in contrast to other functional activities of the system.

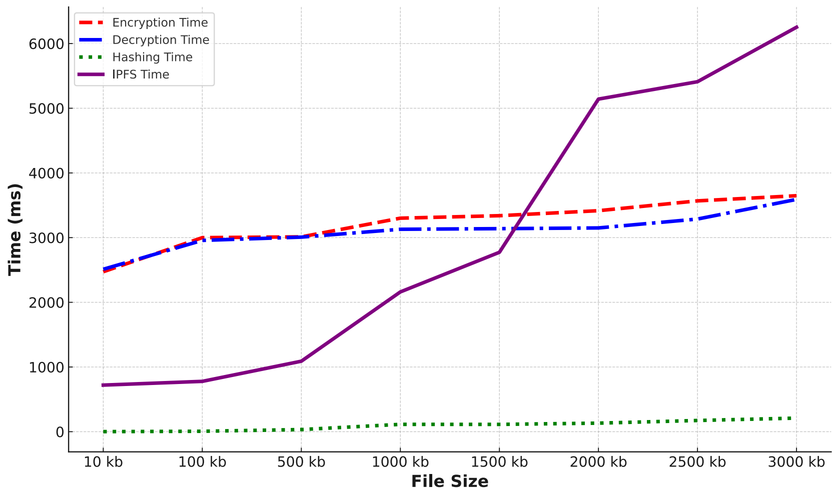


Figure 3. Total time in ms.

5.2.3. Scalability

Finally, we analyze the scalability of the proposed system by measuring the overall throughput of all phases (not only of the encryption one) with respect to different data sizes, ranging from 10 kb to 3000 kb. Figure 4 shows the overall throughput, including data encryption, hashing, and storing the encrypted and hashed data on IPFS. As we can observe from the graph, initially, the throughput was around 3 kb/s for 10 kb of data. However, as data size increases to 3000 kb, the throughput impressively grows to over 307 kilobytes per second. This shows the robust ability of the proposed system to manage and process large data sets effectively, highlighting the proposed solution's scalability. Therefore, the proposed solution is suitable for adoption in data-intensive applications where scalable security is critical.

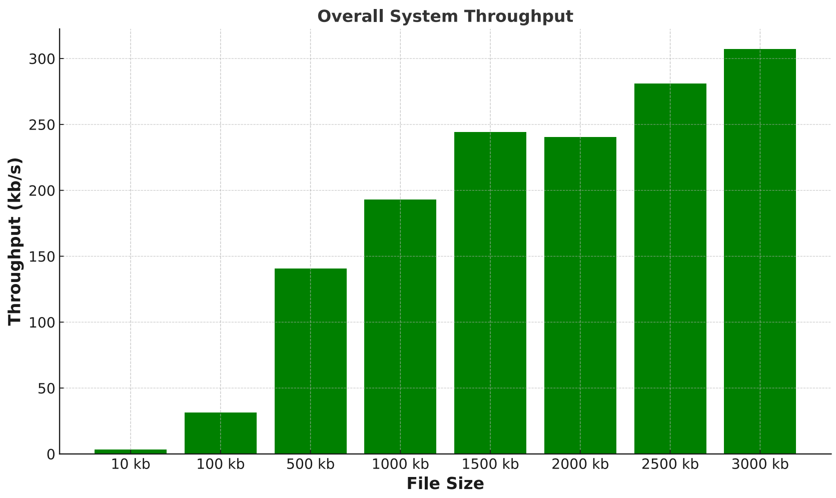


Figure 4. Overall throughput in kb/s.

5.3. Security Evaluation

This section presents a security evaluation of the proposed system. We evaluate the capability of the system to check the data integrity and prevent unauthorized access. The proposed system successfully detected both attacks.

5.3.1. Data Integrity

The primary focus of AES and similar encryption algorithms is to ensure confidentiality. However, these systems do not ensure data integrity, which means that they do not detect data changes during transmission or storage. An attacker could modify encrypted data, which leads to incorrect but successful decryption. Therefore, the AES algorithm requires an additional mechanism for data integrity. The proposed solution addresses this data integrity challenge by integrating a hashing mechanism with encryption. The proposed system computes the original data hash, a unique digital fingerprint of data. After decryption, it computes the hash of decrypted data and compares it with the original hash. If both hashes match, no tampering occurred during the encryption and decryption. The proposed data integrity approach ensures robust security against unauthorized modifications, improving stored or transmitted information security.

5.3.2. Unauthorized Access

The unauthorized access attack consists of accessing the content of the encrypted data without a valid key. We perform an evaluation against this attack, which involves two steps: first, a secure encryption of data with a valid secret key and then an attempt to decrypt these data by using a different, unauthorized secret key. This approach simulates a real-world attack scenario where an attacker tries to gain access to encrypted information. The system successfully identified the unauthorized attempt as it continuously failed to decrypt the data by using the unauthorized key. This resulted in either an empty output or an error. These results show the proposed solution's robust security and efficiency in preventing unauthorized access and the reliability of system encryption and key management schemes in protecting sensitive data.

6. Conclusions

Data immutability, traceability, and availability could be provided by properly integrating blockchain technology and IPFS. This paper proposes a complete architecture that properly combines these two technologies with encryption and hashing algorithms to ensure the mentioned properties efficiently. A set of scalability tests have also been performed to demonstrate the system's capabilities to adapt to an increased amount of data. Finally, some considerations about the prevention of unauthorized access and the ensuring of data integrity are also provided.

Author Contributions: Conceptualization, M.B.S. and S.M.; methodology, M.B.S., S.M. and F.S.; software, M.B.S.; writing—original draft preparation, M.B.S., S.M. and F.S.; writing—review and editing, S.M.; supervision, S.M. and F.S. All authors have read and agreed to the published version of the manuscript.

Funding: This study was partially carried out within the Interconnected Nord-Est Innovation Ecosystem (iNEST) and the initiative "Innovative PhDs which respond to the companies demand of innovation". It received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR)–MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.5–D.D. 1058 23/06/2022 ECS00000043, and MISSIONE 4 COMPONENTE 2, INVESTIMENTO 3.3-DM 352/2022 progetto M4C2 Investimento 3.3). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Data Availability Statement: The source code is available in the public GitHub repository: https://github.com/MuhammadBinSaif/Efficient-and-Secure-Distributed-Data-Storage-and-Retrieval-Using-IPFS-and-Blockchain (accessed on 12 March 2024).

Acknowledgments: During the preparation of this work, the authors used Grammarly's generative AI writing feature and ProWritingAid AI writing assistant in order to check grammar, improve writing, and fix passive verbs to active verbs. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IPFS	Interplanetary File System
CID	Content Identifier
PBKDF2	Password-Based Key Derivation Function 2

References

1. Pouwelse, J.; Garbacki, P.; Epema, D.; Sips, H. The Bittorrent P2P File-Sharing System: Measurements and Analysis. In *Proceedings of the Peer-to-Peer Systems IV*; Castro, M., van Renesse, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 205–216.
2. Mohanta, B.K.; Panda, S.S.; Jena, D. An Overview of Smart Contract and Use Cases in Blockchain Technology. In Proceedings of the 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 10–12 July 2018; pp. 1–4. [CrossRef]
3. Pinto, R.P.; Silva, B.M.C.; Inácio, P.R.M. A System for the Promotion of Traceability and Ownership of Health Data Using Blockchain. *IEEE Access* **2022**, *10*, 92760–92773. [CrossRef]
4. Rana, R.L.; Adamashvili, N.; Tricase, C. The Impact of Blockchain Technology Adoption on Tourism Industry: A Systematic Literature Review. *Sustainability* **2022**, *14*, 7383. [CrossRef]
5. Xia, W.; Chen, X.; Song, C. A Framework of Blockchain Technology in Intelligent Water Management. *Front. Environ. Sci.* **2022**, *10*. [CrossRef]
6. Stockburger, L.; Kokosioulis, G.; Mukkamala, A.; Mukkamala, R.R.; Avital, M. Blockchain-enabled decentralized identity management: The case of self-sovereign identity in public transportation. *Blockchain Res. Appl.* **2021**, *2*, 100014. [CrossRef]
7. Agrawal, T.K.; Kumar, V.; Pal, R.; Wang, L.; Chen, Y. Blockchain-based framework for supply chain traceability: A case example of textile and clothing industry. *Comput. Ind. Eng.* **2021**, *154*, 107130. [CrossRef]
8. Zhu, H.; Zhou, Z.Z. Analysis and outlook of applications of blockchain technology to equity crowdfunding in China. *Financ. Innov.* **2016**, *2*, 29. [CrossRef]
9. Rajasekar, V.; Sondhi, S.; Saad, S.; Mohammed, S. Emerging Design Patterns for Blockchain Applications. In Proceedings of the ICSOFT, Online Event, 7–9 July 2020; pp. 242–249.
10. Bauer, D.P. InterPlanetary File System. In *Getting Started with Ethereum: A Step-by-Step Guide to Becoming a Blockchain Developer*; Apress: Totowa, NJ, USA, 2022; pp. 83–96. [CrossRef]
11. Monrat, A.A.; Schelén, O.; Andersson, K. A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. *IEEE Access* **2019**, *7*, 117134–117151. [CrossRef]
12. ul Haque, A.; Ghani, M.S.; Mahmood, T. Decentralized Transfer Learning using Blockchain & IPFS for Deep Learning. In Proceedings of the 2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 7–10 January 2020; pp. 170–177. [CrossRef]
13. Shah, M.; Shaikh, M.; Mishra, V.; Tuscano, G. Decentralized Cloud Storage Using Blockchain. In Proceedings of the 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI) (48184), Tirunelveli, India, 15–17 June 2020; pp. 384–389. [CrossRef]
14. Zheng, X.; Lu, J.; Sun, S.; Kiritsis, D. Decentralized industrial IoT data management based on blockchain and IPFS. In Proceedings of the IFIP International Conference on Advances in Production Management Systems, Novi Sad, Serbia, 30 August–3 September 2020; Springer: Cham, Switzerland, 2020; pp. 222–229.
15. Huang, H.; Lin, J.; Zheng, B.; Zheng, Z.; Bian, J. When blockchain meets distributed file systems: An overview, challenges, and open issues. *IEEE Access* **2020**, *8*, 50574–50586. [CrossRef]
16. Hassan, M.U.; Rehmani, M.H.; Chen, J. Privacy preservation in blockchain based IoT systems: Integration issues, prospects, challenges, and future research directions. *Future Gener. Comput. Syst.* **2019**, *97*, 512–529. [CrossRef]
17. Satybaldy, A.; Nowostawski, M. Review of techniques for privacy-preserving blockchain systems. In Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure, Taipei, Taiwan, 6 October 2020; pp. 1–9.
18. Wang, S.; Wang, X.; Zhang, Y. A Secure Cloud Storage Framework With Access Control Based on Blockchain. *IEEE Access* **2019**, *7*, 112713–112725. [CrossRef]
19. Hao, J.; Sun, Y.; Luo, H. A safe and efficient storage scheme based on blockchain and IPFS for agricultural products tracking. *J. Comput* **2018**, *29*, 158–167.

20. Arer, M.M.; Dhulavvagol, P.M.; Totad, S. Efficient big data storage and retrieval in distributed architecture using blockchain and ipfs. In Proceedings of the 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 7–9 April 2022; IEEE: Toulouse, France, 2022; pp. 1–6.
21. Hasan, S.S.; Sultan, N.H.; Barbhuiya, F.A. Cloud data provenance using IPFS and blockchain technology. In Proceedings of the Seventh International Workshop on Security in Cloud Computing, Auckland, New Zealand, 8 July 2019; pp. 5–12.
22. Yang, P.; Xiong, N.; Ren, J. Data security and privacy protection for cloud storage: A survey. *IEEE Access* **2020**, *8*, 131723–131740. [CrossRef]
23. Handschuh, H. SHA Family (Secure Hash Algorithm). In *Encyclopedia of Cryptography and Security*; van Tilborg, H.C.A., Ed.; Springer: Boston, MA, USA, 2005; pp. 565–567. [CrossRef]
24. Raeburn, K. Advanced Encryption Standard (AES) Encryption for Kerberos 5. *RFC* **2005**, *3962*, rfc3962. [CrossRef]