symmetry

MDPI

Article

Symmetry in Blockchain-Powered Secure Decentralized Data Storage: Mitigating Risks and Ensuring Confidentiality

Iuon-Chang Lin [1], Yi-Hsuan Kuo [1], Ching-Chun Chang [2], Jui-Chuan Liu [3] and Chin-Chen Chang [3,*]

[1] Department of Management Information Systems, National Chung Hsing University, Taichung 402, Taiwan; iclin@dragon.nchu.edu.tw (I.-C.L.); g110029101@smail.nchu.edu.tw (Y.-H.K.)
[2] Information and Communication Security Research Center, Feng Chia University, Taichung 402, Taiwan; ccc@fcu.edu.tw
[3] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 402, Taiwan; p1200318@o365.fcu.edu.tw
* Correspondence: ccc@o365.fcu.edu.tw

Abstract: In today's digital landscape, the exponential growth of data heightens security risks associated with traditional centralized storage systems. Utilizing blockchain technology, a shift towards decentralized data storage provides a more secure and private alternative. Central to our work is the exploration of symmetry in data management, a concept woven into the fabric of our proposed solution to challenge the inherency in InterPlanetary File System (IPFS) technology. Through the strategic utilization of smart contract-invoked random functions, our blockchain-based solution fragments and securely stores data in order to ensure a symmetrical balance between confidentiality and integrity. Our research endeavors are to contribute a robust, ethically grounded data storage framework fostering advancements in secure data sharing. The implications of this paper are significant in addressing contemporary challenges of data management within the expansive realm of big data.

Keywords: blockchain; ethereum; random number generator; IPFS; smart contracts; data storage

Citation: Lin, I.-C.; Kuo, Y.-H.; Chang, C.-C.; Liu, J.-C.; Chang, C.-C. Symmetry in Blockchain-Powered Secure Decentralized Data Storage: Mitigating Risks and Ensuring Confidentiality. *Symmetry* **2024**, *16*, 147. https://doi.org/10.3390/sym16020147

Academic Editor: Theodore E. Simos

Received: 29 December 2023
Revised: 23 January 2024
Accepted: 24 January 2024
Published: 26 January 2024

1. Introduction

In the current digital era, the amount of data produced and consumed by people has reached astonishing levels. With the continuous development of digital technologies, we have entered a data-driven society where data are widely applied in various fields such as finance, science, healthcare, government, and manufacturing. The value of this data is increasing, as it can be used to support decision making, improve efficiency, and drive innovation. The extensive use of data has greatly accelerated the speed of information dissemination. Through real-time collection, analysis, and transmission of data, people can quickly access the information they need and make more accurate decisions. For example, in the financial sector, high-frequency trading systems use vast amounts of market data to predict fluctuations in stock prices and enable fast trading. In the healthcare field, doctors can diagnose diseases quickly and provide appropriate treatments by analyzing big data.

However, the impact of excessively large amounts of data cannot be ignored. Data issues related to storage, processing, and quality arise. Managing massive data requires significant storage space and powerful computing capabilities. An overload of data can lead to straining information, affecting data availability and making it difficult for individuals to effectively process and interpret the data. The data overloading may result in the inability to identify important data and may cause challenges in analysis and decision making. To address this problem, people have started using cloud computing and big data technologies to store data in cloud servers and process data through distributed computing.

At the same time, the excessive volume of data brings privacy and security risks. Large amounts of personal data are collected and analyzed always. If these data fall into the

hands of malicious attackers or are compromised, it can cause significant losses and pose serious threats to individuals and organizations. Therefore, ensuring data security and privacy has become an urgent issue and people need to develop stricter data protection policies and technological measures to prevent data misuse.

As for data storage systems, traditional centralized management systems often offer relatively high reliability and ease of management. However, they also come with potential risks, especially when facing malicious access or attacks. While centralized management systems have played a significant role in the past, decentralized systems provide better security, flexibility, and trustworthiness in today's complex and ever-changing data environment. Consequently, more and more people are choosing to adopt decentralized data storage systems to meet their needs.

Although traditional centralized data storage systems have been widely used, they are not without drawbacks. Since all data are stored on a single central server or a few central servers, these systems become more vulnerable to attacks. Malicious hackers or insiders, attempting to invade and steal this data, may exploit vulnerabilities or weaknesses, leading to potentially catastrophic consequences including sensitive information leaks, data loss, or system paralysis. Therefore, the use of centralized storage systems faces increasingly severe security risks.

To address centralized systems' security vulnerabilities, more organizations and individuals are shifting to decentralized data storage. Decentralized systems distribute data across multiple nodes to enhance flexibility and redundancy. In this case, data remain safe and the risk of loss is reduced even if a node fails. The distributed storage makes it harder for attackers to target because it requires simultaneous attacks on multiple nodes during intrusion. This decentralization improves data security. Furthermore, decentralized systems foster transparency and trust, preventing unauthorized changes, as every participant can verify data integrity across nodes. Advancements in blockchain and distributed technologies make decentralized systems practical and easier to implement. Emerging data storage solutions embracing blockchain and similar tech can further bolster data security and reliability.

The IPFS (InterPlanetary File System) [1] is a decentralized storage protocol that is widely used today for sharing files and data on a global scale. It has found extensive adoption in various domains and an increasing number of studies are exploring IPFSs as a method for data storage. By utilizing IPFS's decentralized storage approach, this study aims to address the challenges caused by centralized storage systems. In today's digital era, industries across sectors such as agriculture supply chains, finance, healthcare, and industrial control heavily rely on data and require large-scale storage systems [2–5]. Without effective management and security mechanisms, there is a significant risk of malicious manipulation.

The IPFS, as a decentralized file system, offers many advantages but also has some drawbacks. Being a distributed system, the IPFS requires significant communication and data exchanges among nodes and can lead to network latency and performance issues. As the number of nodes increases, the scalability of the system is challenged and the system requires more computational and storage resources to handle large-scale data transmission. Therefore, the performance and scalability of the IPFS remain a challenge. While the IPFS's content-addressable naming ensures data uniqueness and persistence, the resulting hash value will be different, making it difficult to directly access previous versions if a file is altered. This can be problematic for applications that require accurate tracking and management of file versions. Moreover, the IPFS faces issues of node availability and data storage. As data are stored and accessed through decentralized storage nodes in a distributed network, these nodes serve as both data repositories and retrieval points. Since the IPFS relies on node sharing and storage, the corresponding data may become inaccessible if a node is taken offline or shut down. This can be a concern for long-term storage and data persistence. Lastly, security and ethical concerns are crucial for IPFSs. As an open system, anyone can join and store data, potentially leading to the existence of

malicious nodes. While the IPFS provides integrity verification mechanisms based on hash values, it does not entirely prevent data tampering, malicious actions, as well as issues related to data sharing, copyright infringement, illegal content, and ethical concerns. Better protective mechanisms are needed to address these challenges.

The exponential growth of data usage In the digital age has significantly propelled technological and business advancements. While the IPFS brings many conveniences, it also faces challenges in terms of performance, scalability, version management, node availability, security, and ethical considerations. Addressing these challenges requires new technologies and approaches to ensure the security, reliability, and effectiveness of data while extracting valuable insights and information.

For example, the implementation of IPFSs in the agricultural supply chain proposes a method for executing traceable transactions across the agricultural supply chain using the Ethereum blockchain and smart contracts. This approach allows farmers to capture the production cycle status of agricultural products using sensors [2]. By uploading MPEG files to the IPFS, all stakeholders can have real-time access to the products' statuses to eliminate the need for trusted central or intermediary institutions. Smart contracts enable process traceability and provide a reliable transaction history, enhancing data integrity and security. However, this method may encounter challenges due to the current mechanism of uploading images to the IPFS. The process of accessing the data can be time-consuming, potentially affecting data integrity and availability. Additionally, if nodes go offline or experience failures, timely access to the data may be hindered. Consequently, achieving the expected performance benefits may be challenging.

To address the storage efficiency issue, many studies first upload the data to the IPFS and then store the content identifiers (CID) on the blockchain. Steichen et al. [6] proposed a blockchain-based IPFS access control, where the CID of each data block is uploaded to a list in a smart contract, with each CID corresponding to a user's public key. When a user requests data, IPFS nodes verify the access rights based on the list stored on the blockchain. However, large files are divided into numerous file chunks. Managing access control for these data chunks not only consumes a significant amount of transaction time but also reduces the execution efficiency of smart contracts. Moreover, in cases where nodes are offline or experience failures, data may not be promptly accessible and able to gain the expected benefits.

In the context of addressing the challenges posed by centralized and decentralized data storage systems, this research explicitly formulates the following research question: How can a blockchain-based decentralized storage solution enhance the performance, scalability, version management, node availability, and security considerations of existing decentralized storage protocols like IPFSs? We hypothesize that the integration of a blockchain-based decentralized storage solution, leveraging the Ethereum blockchain and smart contracts, will mitigate performance and scalability issues, streamline version management, ensure continuous node availability, enhance data security, and address privacy concerns within the existing IPFS framework. This integration is expected to result in an overall improvement in data availability and reliability, even in the presence of potential node failures, thereby contributing to the advancement of decentralized storage technologies. The hypothesis driving this study is that leveraging the Ethereum blockchain and smart contracts can effectively mitigate these challenges and improve data availability, even when facing potential node failures.

This paper proposes a blockchain-based decentralized storage solution to address IPFS challenges. By leveraging the Ethereum blockchain and smart contracts, it aims to mitigate performance, scalability, version management, node availability, security, and ethical concerns. The decentralized and tamper-resistant nature of blockchain ensures data security and reliability. Verification and consensus mechanisms enhance data integrity, while traceability ensures data provenance. This solution improves data availability to allow timely overcoming despite node failures.

Section 2 describes the background and related works. Section 3 details the proposed system design. The experiment results and analysis are in Section 4. The conclusion completes our study and states the possibility of future work in Section 5.

2. Background and Related Works

2.1. InterPlanetary File System (IPFS)

The InterPlanetary File System (IPFS) is a decentralized peer-to-peer file system that has gone through multiple stages of evolution, gradually improving and expanding its functionalities. The IPFS [1] leverages the advantages of content-addressable protocols found in BitTorrent and Git version control systems. It introduced a distributed hash table (DHT) for data distribution and retrieval. When a file is added to the IPFS, it is divided into multiple small chunks with each assigned a unique hash value called a content identifier (CID). These chunks are stored on nodes' local storage based on their hash values and can be retrieved and accessed using their respective hashes. The method ensures data integrity and verifiability.

CoinList introduced "Filecoin", a decentralized storage platform based on the IPFS, in September 2018. It offers secure storage and transmission using encrypted cryptocurrency. Filecoin connects users and providers through an incentive mechanism, where users pay with Filecoin and providers receive rewards. Leveraging IPFS's distributed storage technology, Filecoin ensures efficient file storage and transmission and enhances network security and reliability.

The IPFS introduced content routing, which enabled nodes to locate data using hash values without knowing their physical locations. This allows direct peer-to-peer data transmission between nodes, eliminating the need for centralized servers.

Another important development is the InterPlanetary Name System (IPNS) [7], which provides a mechanism to map mutable IPFS addresses to human-readable names. The IPNS utilizes encryption techniques from public key infrastructure (PKI) by binding private keys to specific names to ensure that only the holders of the private keys can update the names. This allows users to access and share data using names without concern about address changes.

The IPFS has a wide range of applications. It can be used for decentralizing storage, sharing data, creating decentralized websites and applications, and enabling historical version tracking of data. The decentralized nature of the IPFS and its data integrity protection make it a powerful tool for combating data censorship and achieving a decentralized internet. It addresses the risk of single point of failure associated with centralized servers, while also increasing data redundancy and reliability. Additionally, IPFS's peer-to-peer transmission and content routing capabilities facilitate faster and more efficient data transfer. The IPFS supports file version control and data validation through unique hash values. Users can track and manage different file versions easily. Hash value comparisons ensure file integrity and detect tampering.

The IPFS differs from traditional HTTP (hypertext transfer protocol) in how files are located and transmitted. Instead of using domain names and IP addresses, the IPFS uses unique hash values to identify file locations. This decentralized approach, in which IPFS nodes search and transmit data using routing protocols, eliminates the single point of failure found in centralized servers and improves reliability and scalability.

IPFSs, as an emerging technology, may raise various concerns and considerations regarding data integrity, confidentiality, and ethical implications in their applications. These issues include the following: (a) the IPFS is an open system which means that there may be illegal or controversial content in the IPFS network, such as pirated data, pornography, or hate speech. Such uncontrolled content can raise moral concerns and require discussions and solutions regarding the legality and morality of the content. (b) Important data uploaded to the IPFS without encryption may be targeted by malicious actors aiming to crack the CID to access sensitive information, risking privacy breaches. (c) Concerns arise regarding piracy and unauthorized sharing of copyrighted materials, potentially impacting

creative industries and intellectual property holders negatively. (d) As the data volumes increase, the current adoption of IPFSs remains limited, which could lead to performance degradation and latency issues within the IPFS network.

To address these concerns and drawbacks, it is crucial to consider ethical implications when using IPFSs. To implement content moderation mechanisms, enhancing privacy protection measures and strengthening copyright safeguards are important solutions. Users should also handle and share content responsibly to adhere to legal and ethical requirements.

This paper aims to address these existing issues by leveraging the mechanisms and operational models of blockchain technology to enhance data integrity, confidentiality, and ethical considerations. Increasingly, applications are encrypting data before uploading it to the IPFS and storing the CID using blockchain mechanisms. Thus, this paper also endeavors to improve the mechanism of data preservation by exploring whether blockchain alone can serve these purposes and address the current limitations of IPFSs.

2.2. Applications of the InterPlanetary File System (IPFS)

Some decentralized systems utilize IPFSs to handle large data volumes. In an agricultural supply chain study [2], farmers capture crop images using sensors and upload them to the IPFS, as depicted in Figure 1. Stakeholders trace transactions and information through smart contracts, ensuring transparency and mitigating risks. Similarly, in another paper [3], a public blockchain-based invoice financing platform stores encrypted data in an IPFS and hash values in smart contracts. This enhances transparency and reduces risks for financial institutions.

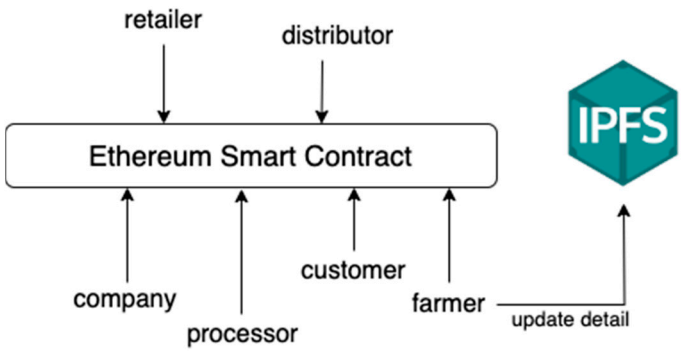

Figure 1. IPFS application diagram 1.

In [4], the paper highlights the importance of electronic health records as crucial evidence for healthcare institutions and pharmaceutical companies. To overcome the single point of failure issue in the current centralized management of electronic health records, the paper proposes encrypting and uploading medical data to the IPFS. It ensures the privacy of patients' data and unauthorized individuals are unable to access it. The architecture is depicted in Figure 2. Similarly, [5] focuses on the application of IoT and presents a work centered around decentralized identity verification and distributed data storage using an IPFS and blockchain mechanisms.

The cited literature discusses various applications of IPFSs and highlights potential challenges and ethical concerns. Issues include the public visibility of data, privacy risks, performance limitations, and latency. To address them, solutions such as content moderation, privacy measures, and copyright protection are proposed. This research aims to present an optimized decentralized storage architecture.

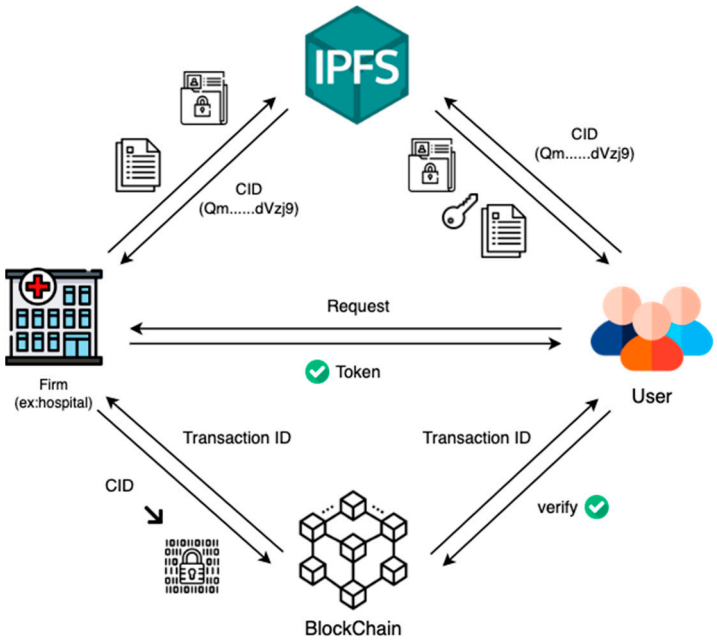

Figure 2. IPFS application diagram 2.

3. System Design

Data storage architecture organizes and manages data, ensuring confidentiality, integrity, and availability. An optimal architecture safeguards against attacks, misuse, and unauthorized alterations. This paper focuses on comparing the decentralized architecture of IPFSs with our research, aiming to achieve IPFS-like results using only blockchain and optimizing its performance.

3.1. System Overview

Our proposed framework addresses the challenges of the IPFS [8], a decentralized data storage and transmission system. By utilizing smart contracts on the Ethereum blockchain, we ensure secure and efficient data sharing. In our architecture, smart contracts store the original data and generate access seeds using transaction information. Our solution not only prioritizes data availability, privacy, and ethical considerations but also mitigates the risks associated with uploading unencrypted data in the IPFS. With our approach, we enhance data security and control to protect against malicious exploitation and unauthorized access. Figure 3 provides an overview of the proposed decentralized data storage solution's system architecture. The main entities involved include firms (illustrated here with string uploads), users, and blockchain-executing smart contracts. Furthermore, each participant in the blockchain must possess an Ethereum account with a unique Ethereum address (EA) for identification purposes.

Each participating entity has a mutual relationship with the smart contract, and their roles are summarized as follows:

1. The firm can upload original data to the blockchain (using TXT files as an example) using smart contracts. The data on the blockchain are transformed, split, shuffled, and randomized with a unique seed for access control.
2. Data consumers who wish to access the content can request the seed (a key or reference) from the enterprise or entity. After verification, the seed will be provided.
3. Only those with the seed can use smart contracts to access and decrypt the original data. It ensures data security and controlled access.

In the blockchain, all data and records are digitally signed and attributed to a specific participant. This means that the entity uploading the data (in this case, the firm uploading the TXT file) is indisputably the owner of such actions and is responsible for any inaccuracies or fraudulent documents.

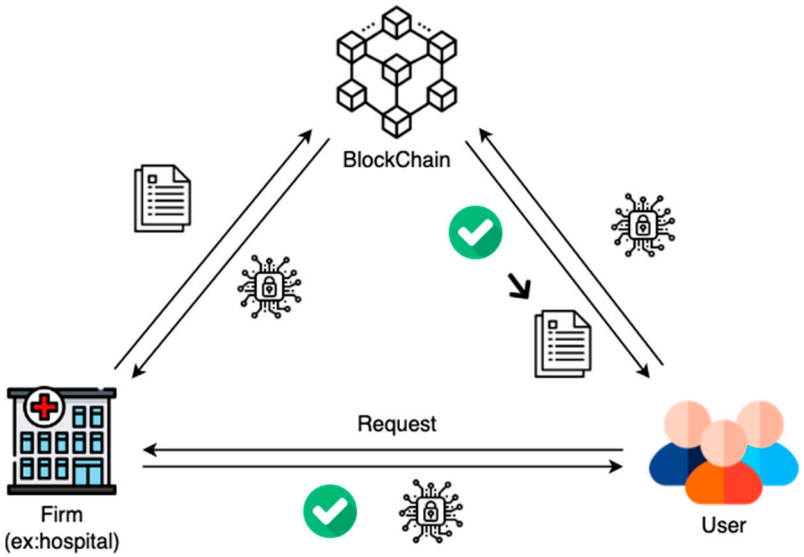

Figure 3. Secure decentralized data storage architecture based on Ethereum.

3.2. Secure Decentralized Data Storage Architecture Based on Ethereum

Each participant has an Ethereum address (EA) and interacts by invoking functions in the smart contract. Figure 4 shows a sequence diagram of a firm uploading data and users accessing it. The diagram depicts the process of the company shuffling and storing the data on the blockchain, as well as generating and storing the seed. Data requesters obtain the seed from the company and use it to restore the data. This process ensures that only authorized users can access and restore the original data. The following chapter will provide a more detailed introduction to the three main steps and their algorithms.

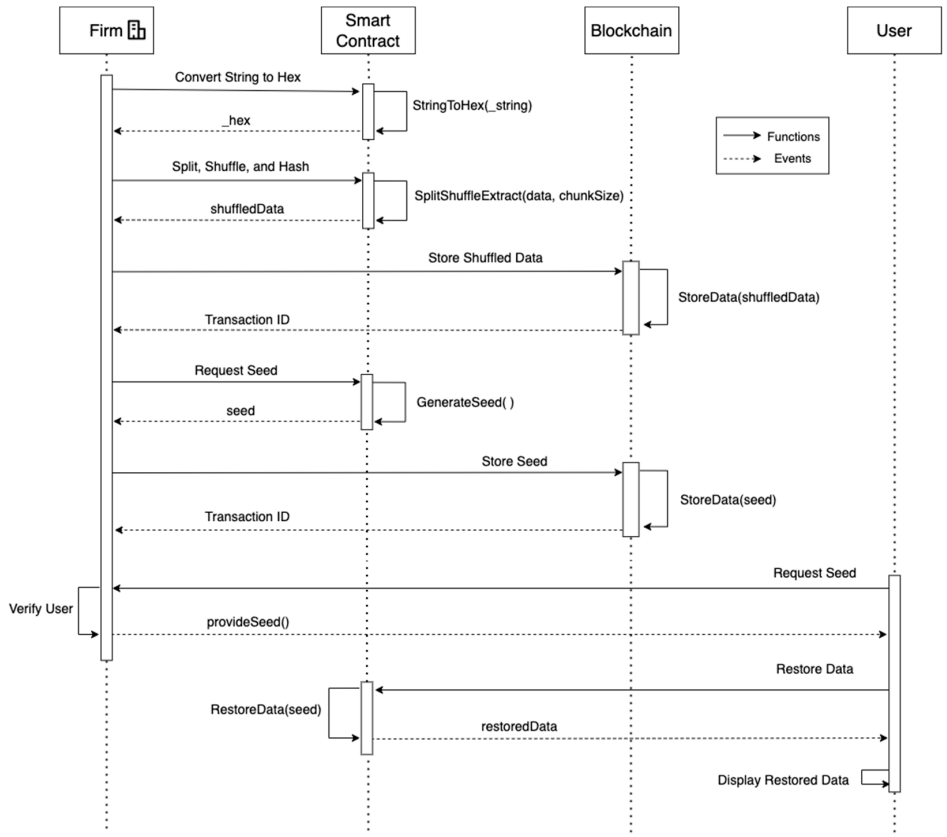

Figure 4. Sequence diagram.

The sequence diagram in Figure 5 illustrates the process of a firm uploading data to the chain by converting the data into an executable format for the smart contract. As depicted in Figure 5, the firm sends a request to the smart contract, specifically the Convert String to Hex() request, to convert a string into a hexadecimal format. Upon receiving the request, the smart contract executes the StringToHex(_string) function, converting the string into hexadecimal format _hex, and returns it to the firm.

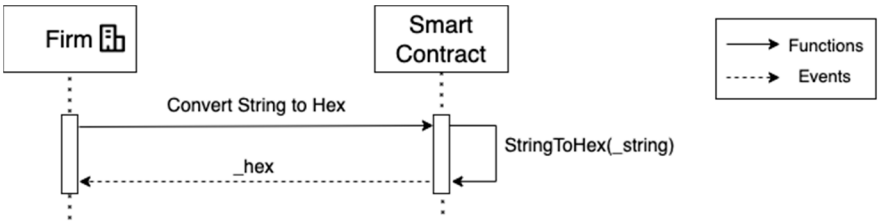

Figure 5. Sequence diagram of converting the data into an executable format.

The sequence diagram in Figure 6 illustrates the firm's process of splitting, randomly shuffling, and hashing the data, as well as storing the relevant information on the blockchain. In the diagram, the firm sends a request to the smart contract, asking for the operations of data splitting, shuffling, and hashing. The smart contract internally invokes the function SplitShuffleExtract(), which handles the data by splitting, shuffling, and hashing it, and then returns the shuffled data, referred to as shuffledData, to the firm.

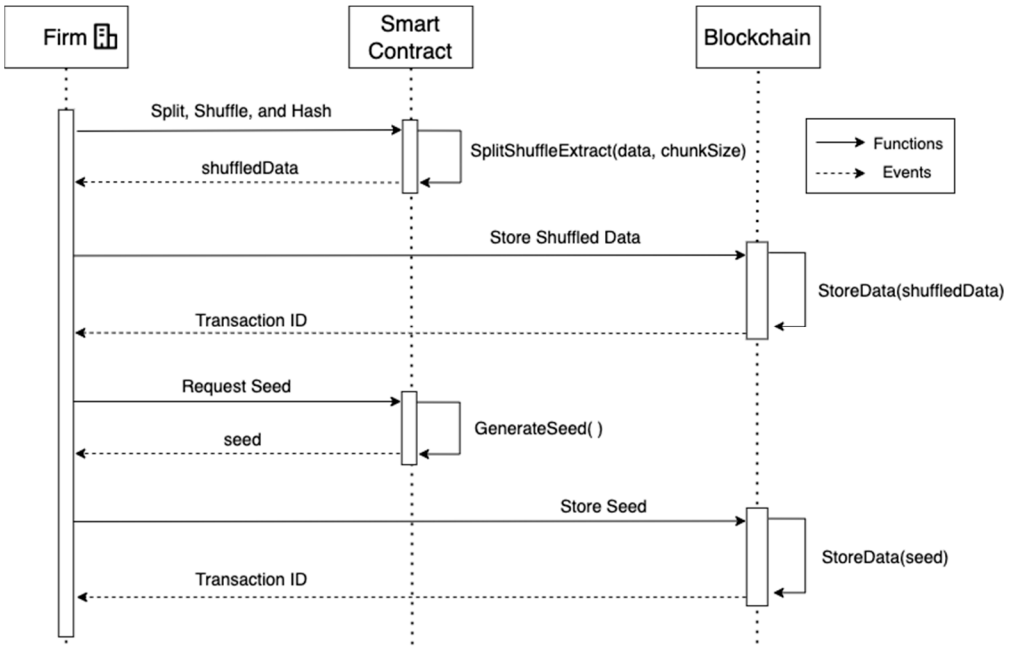

Figure 6. Sequence diagram of the firm's process of splitting, randomly shuffling, hashing the data, and storing seed.

Next, the firm stores the shuffled data on the blockchain by sending a request, StoreShuffledData(), to the blockchain. The blockchain, in turn, invokes the function StoreData(), which is responsible for storing the shuffled data on the blockchain. The result of the storage operation, represented by the transaction ID, is returned to the firm.

Subsequently, the firm requests the smart contract to generate a seed by sending a Request Seed. The smart contract internally calls the function GenerateSeed(), which generates a seed and returns it to the firm. Finally, the firm stores the generated seed on the blockchain by sending a request, StoreSeed(), to the blockchain. The blockchain, again

using the function StoreData(), stores the seed on the blockchain, and the result of the storage operation (Transaction ID) is returned to the firm.

In summary, this sequence diagram demonstrates the firm's operations of data splitting, shuffling, and hashing, as well as storing the shuffled data and the generated seed on the blockchain. The process involves several internal functions, including SplitShuffleExtract(), GenerateSeed(), and the blockchain's StoreData() function.

The sequence diagram in Figure 7 depicts the process in which a user requests a seed from the firm and then uses the seed to restore the data. In the diagram, the user sends a "Request Seed()" request to the firm, seeking to obtain a seed. The firm internally verifies the user's identity to ensure authorization. Once the firm confirms the user's identity, it provides the seed by invoking the function "provideSeed()" for the user.

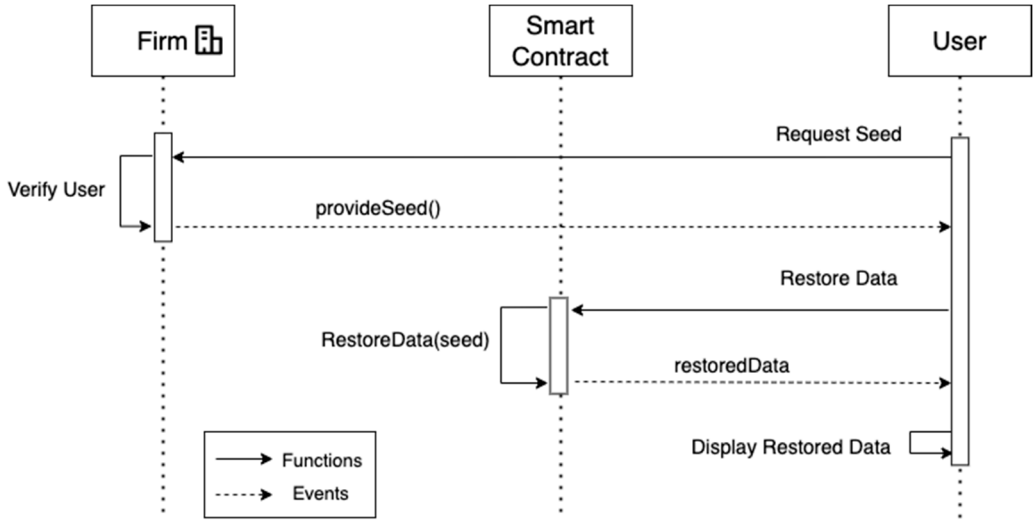

Figure 7. Sequence diagram of the request seed and restore data function.

Next, the user sends a request to the smart contract for data restoration. The smart contract internally calls the function "RestoreData()" to restore the data using the provided seed. The smart contract then returns the restored data to the user. Finally, the user can display and access the restored data for subsequent use.

The sequence diagram combines secure data handling, decentralized blockchain storage, and seed utilization for secure and traceable data storage. Only authorized users can access and restore data to enhance security and privacy. Storing data and seeds on the blockchain ensures immutability and traceability. This framework effectively addresses integrity, confidentiality, and ethical concerns solely through the utilization of the blockchain.

4. Experimental Results and Analysis

This chapter explores existing decentralized data storage frameworks and compares their security analysis with the proposed framework. It discusses the algorithms defining the working principles of the three-phase method presented in the previous chapter.

Our research framework, depicted in Figure 3, will be thoroughly discussed for its security, confidentiality, availability, and integrity, along with the underlying reasons. We convert data into a format processed by smart contracts, performing splitting, random shuffling, and hashing to protect data confidentiality. Storing data on the blockchain provides security, as it is immutable and cannot be modified or deleted. The generated seed serves as a crucial key accessible only to authorized users and enhances data security. Multiple nodes maintain identical data copies to ensure consistency through consensus verification.

Overall, this process ensures data confidentiality for authorized users, data availability through reconstruction algorithms, and data consistency across different environments. The

use of blockchain technology enhances data confidentiality, consistency, and availability with its immutable and decentralized storage capabilities.

4.1. Implementation

In our implementation, the architecture design and the corresponding sequence diagram guide the execution through three distinct phases. The first phase focuses on the crucial task of converting data into a hexadecimal format and lays the foundation for subsequent secure processing.

Within this phase, we have meticulously implemented the functionality through the creation of the "StringToHex()" function, as shown in Algorithm 1. This function takes a user-provided string as input and transforms it into its hexadecimal representation, as outlined in Algorithm 1. The conversion process is fundamental and plays a pivotal role in ensuring data confidentiality, availability, and consistency.

Furthermore, the algorithm can be expanded to accommodate a wider range of media formats. This includes incorporating functionality to handle diverse data types such as images, audio, video, and other binary formats. By doing so, we aim to make our framework more versatile to address the broader spectrum of data that users may encounter in real-world scenarios. The extension to support richer media formats not only enhances the framework's applicability but also makes it more inclusive for various data storage and retrieval needs. Digital identity for digital city operating system [9], biometric identity management [10], and large-scale Internet of Things [11] are some of the example fields in which the algorithm can be applied.

The algorithm design goes beyond a simple conversion and incorporates essential steps such as data splitting, shuffling, hash assignment, and seed retrieval. This multifaceted approach contributes to a robust framework that not only addresses data integrity but also ensures data security during storage and restoration. The foundational step is essential as it establishes the groundwork to ensure data confidentiality, availability, and consistency in subsequent phases.

Algorithm 1 String to Hex Conversion

```
function STRINGTOHEX(string)
    bytes ← convert_string_to_bytes(string)
    hex ← new_bytes(2 × length(bytes))
    for i ← 0 to length(bytes) do
        char ← bytes[i]
        byte1 ← convert_to_hex_byte1(char)
        byte2 ← convert_to_hex_byte2(char)
        hex[2 × i] ← byte1
        hex[2 × i + 1] ← byte2
    return hex
```

Moving into the second phase, we delve into the intricacies of data manipulation through the implementation of the "SplitShuffleExtract()" function. This function, highlighted in Algorithm 2, orchestrates the processes of data splitting, shuffling, hash value assignment to different blocks, and seed extraction. By dividing the original data into fixed-sized blocks, and shuffling and hashing the blocks, this phase not only enhances data security but also extracts the necessary seed for subsequent storage and restoration processes. In Algorithm 2, the symbol |= means that data are appended to the last dataChunk, and the function returns shuffledData for the next phase.

Algorithm 2 Data Split, Shuffle, and Seed Extraction

```
function SplitShuffleExtract(data, chunkSize)
    dataHex ← StringToHex(data)
    dataBytes ← convert to bytes(dataHex)
    dataSize ← length(dataBytes) / chunkSize
        shuffledData ← new array of bytes32
        for i ← 0 to dataSize do
            dataChunk ← new bytes32
            for j ← 0 to chunkSize do
                char ← convert to byte(dataBytes[i × chunkSize + j])
                dataChunk |= convert to bytes32(uint256(char) ≪ (8 × j))
            index ← hash and allocate(dataChunk, i) % dataSize
            shuffledData[index] ← dataChunk
    return shuffledData
```

The final phase of our implementation centers around secure data storage and restoration, involving interactions with both the blockchain and users. We store the shuffled data on the blockchain to ensure its security and immutability. Simultaneously, we save the generated seed on the blockchain as a key for data restoration. When other users need access to the original data, they can request the seed from the data owner. To guarantee secure communication between users, a thorough verification and authorization process is implemented. Algorithm 3 outlines the steps involved in utilizing the seed for data restoration to ensure data confidentiality, availability, and consistency throughout the process.

Algorithm 3 Data Restoration

```
function RestoreData
    restoredData ← new bytes(dataSize)
    for i ← 0 to dataSize do
        dataChunk ← shuffledData[i]
        originalByte ← convert to byte(dataChunk ≫ (seed mod 32))
        restoredData[i] ← originalByte
    return convert to string(restoredData)
```

As we progress through the subsequent phases, these initial measures collectively contribute to a secure and resilient data processing pipeline to ensure that our implementation aligns with the core principles of data integrity, confidentiality, and traceability. The detailed integration of these functionalities and our commitment to a thorough and well-thought-out approach to addressing the complexities associated with data handling are demonstrated within our proposed system.

Our experiments were conducted using the Ganache as the designated testing environment. Two different-sized text files were employed to assess the performance of our system. The first file, with a size of 1 K, exhibited an execution time of 4678 milliseconds, as illustrated in Figure 8. Meanwhile, the second file with a larger size of 8 K demonstrated a slightly increased execution time of 5050 milliseconds and is depicted in Figure 9.

Regarding the transaction fees for the execution of smart contracts, we took the example of 1 K sized data, which consumed 4,849,468 gas, calculated at 20 gwei per gas, resulting in a total transaction cost of ETH 0.09698936. For 8 K sized data, it consumed 4,628,148 gas, calculated at 20 gwei per gas, resulting in a total transaction cost of ETH 0.09256296.

```
Using network 'development'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.


  Contract: DataShuffling
Small Data - Execution time: 4678 ms
Small Data - Gas used: 4849468
Small Data - Gas price: 20000000000 wei
Small Data - Cost in ETH: 0.09698936
    ✔ should measure execution time for different data sizes (4713ms)


  1 passing (5s)
```

Figure 8. Execution time of the file with a size of 1 K.

```
Using network 'development'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.


  Contract: DataShuffling
Medium Data - Execution time: 5050 ms
Medium Data - Gas used: 4628148
Medium Data - Gas price: 20000000000 wei
Medium Data - Cost in ETH: 0.09256296
    ✔ should measure execution time for different data sizes (5091ms)


  1 passing (5s)
```

Figure 9. Execution time of the file with a size of 8 K.

We also provide detailed information about deploying the smart contract to the blockchain shown in Figures 10 and 11. The contract deployment consumed 535,306 gas, calculated at 20 gwei per gas, resulting in a total transaction cost of ETH 0.01070612.

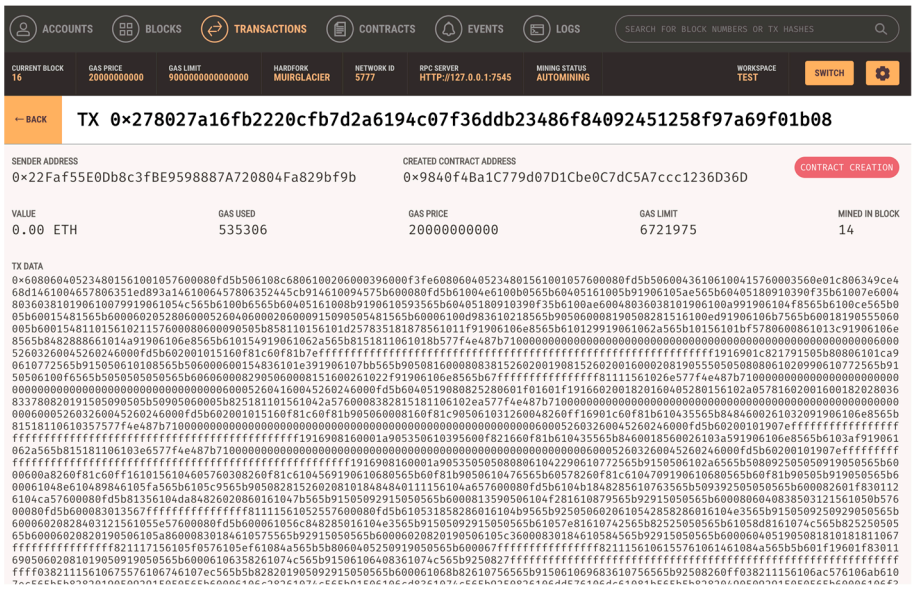

Figure 10. Transaction data of contract creation viewed on Ganache.

```
2_deploy_contracts.js
=====================

   Replacing 'DataShuffling'
   -------------------------
   > transaction hash:    0x278027a16fb2220cfb7d2a6194c07f36ddb23486f84092451258f97a69f01b08
   > Blocks: 0            Seconds: 0
   > contract address:    0x9840f4Ba1C779d07D1Cbe0C7dC5A7ccc1236D36D
   > block number:        14
   > block timestamp:     1705429247
   > account:             0x22Faf55E0Db8c3fBE9598887A720804Fa829bf9b
   > balance:             95.86082044
   > gas used:            535306 (0x82b0a)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.01070612 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.01070612 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.01070612 ETH
```

Figure 11. The detail of transaction data of contract creation.

4.2. Security Analysis

Information security entails safeguarding information and information systems against unauthorized access, use, disclosure, destruction, alteration, viewing, recording, and disposal.

- Confidentiality

This study proposes multiple security measures to ensure confidentiality. By employing data transformation and shuffling techniques, we convert the data into an unintelligible hexadecimal format and safeguard it from unauthorized access. Utilizing blockchain for data storage and verification prevents unauthorized tampering and access. Verification and authorization mechanisms restrict access to sensitive data and enhance confidentiality. These measures collectively protect data from unauthorized disclosure.

- Integrity

Through the processes of data splitting, shuffling, and hashing, we ensure effective protection and handling of data before storing it on the blockchain. These processes not only guarantee data integrity but also detect any potential tampering or data errors. The immutability of the blockchain and the use of digital signatures are employed to prevent tampering. These security measures ensure the long-term integrity and reliability of the data.

- Availability

By utilizing blockchain technology, data storage and access become decentralized and distributed, thereby improving system availability. Additionally, users can directly interact and share data without relying on central authorities, enhancing data availability and sharing. The system's scalability and decentralized nature further ensure data availability and reliability.

- Authentication

To ensure authentication, this study employs a verification mechanism between the firm and the user. This prevents unauthorized access and ensures system security and reliability.

- Access Control

To enforce access control, this study utilizes a multi-layered mechanism where users must undergo authentication and authorization to request the seed and access the original data. It ensures that only authorized and verified users can manipulate the data and safeguard the system from unauthorized access, maintaining its security and reliability.

- Nonrepudiation

The design of this study incorporates a mechanism called nonrepudiation, which prevents denial of actions by both the firm storing data on the blockchain and the user accessing and using the data. This mechanism ensures trust and traceability in user interactions by leveraging the immutability and transaction records of blockchain technology. It provides legal and regulatory assurances, as neither party can deny their actions once the data are stored and accessed.

- Comparisons

This subsection compares the security analysis of the proposed framework with the others in the literature reviews shown in Table 1. Unlike other applications, our framework randomly stores data on the blockchain instead of directly uploading it to a shared P2P network like the IPFS. This ensures that the original data remain private and can only be restored using a seed. The framework includes identity verification and access control mechanisms, providing strict access control for authorized users. Leveraging blockchain technology achieves nonrepudiation and prevents users from denying their actions of storing and accessing data. In contrast to IPFS applications, where data can be accessed with the CID, our framework offers enhanced privacy protection.

Table 1. Security analysis comparisons.

	[2]	[3]	[4]	[5]	Ours
Confidentiality		V	V		V
Integrity			V	V	V
Availability					V
Authentication			V	V	V
Access Control			V	V	V
Nonrepudiation	V	V	V	V	V

This framework solely utilizes the blockchain to store shuffled data and seeds, improving system efficiency and scalability for different data processing algorithms in various scenarios. In comparison, IPFS applications rely on a P2P network for data storage and sharing, which may be constrained by network connectivity and node availability. While there are advantages to integrating blockchain and IPFSs, it also presents its own set of challenges.

5. Conclusions and Future Works

This paper presents a blockchain-based secure data storage architecture to enhance security and privacy protection. Our contributions lie in two key aspects:

Firstly, we propose a system architecture that allows users to securely store data on the blockchain, ensuring privacy and data integrity through smart contracts and encryption mechanisms. Next, we introduce data shuffling techniques involving segmentation, shuffling, and hashing, adding an extra layer of security that makes it difficult to reverse engineer or analyze the data, even when stored on the blockchain.

Our research emphasizes user autonomy and control, respecting data ownership and privacy rights. Users have the authority to determine data sharing scopes and conditions and can revoke permissions or restrict access as needed. The implemented blockchain-based secure data storage architecture provides confidentiality, integrity, traceability, identity verification, and access control. Additionally, it addresses ethical considerations, ensuring compliance with moral and ethical principles.

In conclusion, this study's contributions to secure data storage and privacy protection have significant implications for the field, providing valuable insights for future research

in data management and privacy protection. The proposed blockchain-based secure data management approach in this study opens up several potential future research directions.

Future research can explore more efficient algorithms and techniques to improve the performance and processing speed of data handling and blockchain operations.

- System scalability: To accommodate increasing data volume and user numbers, future studies can focus on scalable approaches for large-scale data management, user demands, and diverse data types.
- Enhanced security: Further research can investigate advanced data shuffling techniques, identity verification mechanisms, and access control strategies to strengthen the security of the proposed system.

Author Contributions: Conceptualization, I.-C.L. and C.-C.C. (Chin-Chen Chang); methodology, I.-C.L.; software, Y.-H.K.; validation, I.-C.L., Y.-H.K., C.-C.C. (Ching-Chun Chang), J.-C.L. and C.-C.C. (Chin-Chen Chang); formal analysis, I.-C.L. and Y.-H.K.; investigation, Y.-H.K.; resources, I.-C.L.; data curation, Y.-H.K.; writing—original draft preparation, I.-C.L. and Y.-H.K.; writing—review and editing, C.-C.C. (Ching-Chun Chang), J.-C.L. and C.-C.C. (Chin-Chen Chang); visualization, I.-C.L.; supervision, I.-C.L. and C.-C.C. (Chin-Chen Chang). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

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

References

1. Benet, J. IPFS—Content addressed versioned P2P file system. *arXiv* **2014**, arXiv:1407.3561.
2. Salah, K.; Nizamuddin, N.; Jayaraman, R.; Omar, M. Blockchain-Based Soybean Traceability in Agricultural Supply Chain. *IEEE Access* **2019**, *7*, 73295–73305. [CrossRef]
3. Guerar, M.; Merlo, A.; Migliardi, M.; Palmieri, F.; Verderame, L. A fraud-resilient blockchain-based solution for invoice financing. *IEEE Trans. Eng. Manag.* **2020**, *67*, 1086–1098. [CrossRef]
4. Sun, J.; Yao, X.; Wang, S.; Wu, Y. Blockchain-Based Secure Storage and Access Scheme for Electronic Medical Records in IPFS. *IEEE Access* **2020**, *8*, 59389–59401. [CrossRef]
5. Dwivedi, S.K.; Amin, R.; Vollala, S. Blockchain-Based Secured IPFS-Enable Event Storage Technique with Authentication Protocol in VANET. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1913–1922. [CrossRef]
6. Steichen, M.; Fiz, B.; Norvill, R.; Shbair, W.; State, R. Blockchain-based decentralized access control for IPFS. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018.
7. Cristea, A.G.; Alboaie, L.; Panu, A.; Radulescu, V. Offline but still connected with IPFS based communication. *Proc. Comput. Sci.* **2020**, *176*, 1606–1612. [CrossRef]
8. Muralidharan, S.; Ko, H. An InterPlanetary file system (IPFS) based IoT framework. In Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 11–13 January 2019.
9. Asamoah, K.O.; Xia, H.; Amofa, S.; Amankona, O.I.; Luo, K.; Xia, Q.; Gao, J.; Du, X.; Guizani, M. Zero-Chain: A Blockchain-Based Identity for Digital City Operating System. *IEEE Internet Things J.* **2020**, *7*, 10336–10346. [CrossRef]
10. Salem, S.H.G.; Hassan, A.Y.; Moustafa, M.S.; Hassan, M.N. Blockchain-based biometric identity management. *Clust. Comput.* **2023**. [CrossRef]
11. Xiong, R.; Ren, W.; Hao, A.; He, J.; Choo, K.K.R. BDIM: A Blockchain-Based Decentralized Identity Management Scheme for Large Scale Internet of Things. *IEEE Internet Things J.* **2023**, *10*, 22581–22590. [CrossRef]