Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.051612

ARTICLE

Designing a Secure and Scalable Data Sharing Mechanism Using Decentralized Identifiers (DID)

Iuon-Chang Lin[1], I-Ling Yeh[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

[2]Information and Communication Security Research Center, Feng Chia University, Taichung, 407, Taiwan

[3]Department of Information Engineering and Computer Science, Feng Chia University, Taichung, 407, Taiwan

*Corresponding Author: Chin-Chen Chang. Email: ccc@o365.fcu.edu.tw

Received: 10 March 2024 Accepted: 31 May 2024 Published: 20 August 2024

ABSTRACT

Centralized storage and identity identification methods pose many risks, including hacker attacks, data misuse, and single points of failure. Additionally, existing centralized identity management methods face interoperability issues and rely on a single identity provider, leaving users without control over their identities. Therefore, this paper proposes a mechanism for identity identification and data sharing based on decentralized identifiers. The scheme utilizes blockchain technology to store the identifiers and data hashed on the chain to ensure permanent identity recognition and data integrity. Data is stored on InterPlanetary File System (IPFS) to avoid the risk of single points of failure and to enhance data persistence and availability. At the same time, compliance with World Wide Web Consortium (W3C) standards for decentralized identifiers and verifiable credentials increases the mechanism's scalability and interoperability.

KEYWORDS

Self-sovereign identity; decentralized identifier; blockchain

1 Introduction

With the development of digitalization, more and more data are converted from physical paper records to digital storage. The digitized data can often be shared by multiple entities. Therefore, how to securely store and share data while ensuring privacy has become an important issue.

Currently, many traditional businesses still rely on centralized architectures to store and manage identity data. Although centralized storage systems are easier to implement and manage, they also increase the risk of data leakage, making them more susceptible to hacker attacks and leading to large-scale data leaks. In 2018, Cambridge Analytica illegally obtained the personal data of Facebook users through a psychological test application [1]. In 2019, Twitter allowed advertisers to access users' email addresses and phone numbers for targeted advertising due to an internal error [2]. Additionally, centralized systems often fail to provide a flexible, manageable, and scalable access mechanism and limit the potential applications of such systems.

Therefore, researchers have begun exploring decentralized identity management and data storage solutions. Although early models of self-sovereign identity (SSI) gave users control over their identities, but lacked a unified identity standard, leaving room for optimization in terms of scalability, interoperability, and compatibility. To address this issue, the World Wide Web Consortium (W3C) proposed a framework for verifiable credentials [3] in May 2017 and released DID 1.0 [4] in December 2019. The emergence of this standard has made using decentralized identifiers (DIDs) to achieve self-sovereign identity a new research direction.

The rationale behind using blockchain for data storage often revolves around its key features such as decentralization, immutability, and transparency. We elaborate on each as follows:

Decentralization: Traditional data storage methods often rely on central servers or databases controlled by a single entity. This centralized approach can be prone to single points of failure, censorship, and data breaches. Blockchain, however, operates on a decentralized network of nodes, where each participant holds a copy of the entire ledger. This distributes the control and responsibility for data storage across the network, making it more resilient and secure.

Immutability: Once data is recorded on a blockchain, it becomes virtually impossible to alter or delete without the consensus of the network participants. Each block in the chain contains a cryptographic hash of the previous block, creating a tamper-evident record of transactions. This immutability provides strong assurances about the integrity and authenticity of the stored data, which is particularly valuable in scenarios where data tampering is a concern.

Transparency: Blockchain technology enables transparent and auditable data storage. All transactions are recorded on a public ledger, and visible to anyone with access to the network. This transparency fosters trust among participants by providing visibility into the history of data transactions and ensuring accountability.

Security: Blockchain networks use cryptographic techniques to secure data and transactions. Consensus mechanisms such as proof of work or proof of stake ensure that malicious actors cannot easily manipulate the data or disrupt the network. Additionally, the distributed nature of blockchain reduces the risk of data loss due to localized failures or cyberattacks.

Use Cases: Blockchain is particularly well-suited for applications where multiple parties need to share and trust data without relying on a central authority. Examples include supply chain management, identity verification, healthcare records management, and financial transactions.

While blockchain offers unique advantages for data storage, it is essential to acknowledge its limitations and consider the suitability of alternative methods for specific use cases. Factors such as scalability, energy consumption, and regulatory considerations may influence the decision to adopt blockchain technology. Ultimately, the selection of a data storage solution should be guided by a thorough assessment of the requirements, risks, and benefits involved.

In this paper, we propose a decentralized identity and data-sharing framework following the standards proposed by W3C. Through decentralized identifiers, the framework achieves de-identification of identity data and increases scalability. Based on the distributed, public, and immutable nature of blockchain [5], it ensures data integrity and availability. By storing data on InterPlanetary File System (IPFS) [6] and only uploading the hash and corresponding DID to the chain, the framework effectively reduces storage costs while ensuring data availability, validity, and correctness.

2 Related Works

With the development of decentralized technology, there is an increasing implementation of digital identity and data storage in a decentralized manner enhancing the autonomy of digital identity and the security of data storage.

Naik et al. [7] proposed a decentralized framework using Ethereum [8] and smart contracts for patient consent management, allowing patients to control their electronic health records and addressing privacy, and data tampering issues in traditional centralized systems. However, storing all identity data on the blockchain incurred high costs on the Ethereum blockchain and made it challenging to implement large-scale data management.

Bhandari et al. [9] introduced a distributed medical healthcare record management system based on blockchain technology to ensure the security and privacy of data sharing. Users register upon the first access and store their identity data in the backend application. The access control is implemented through smart contracts which enable users to securely access, store, and share patient medical records. Moreover, this system utilizes IPFS to store documents, creating a permanent and decentralized storage method. However, this model only distributes data storage, but users' data is still controlled by the central entity.

Otta et al. [10] proposed an identity management solution for Indian migrant workers Users register their data, and the system creates a new user profile on the blockchain to store the user's data and identifier on the blockchain. It solves the identity identification issues faced by migrant workers and provides them with a safer and more reliable identity management mechanism. However, this approach to all personal information on the chain may lead to high costs, and the lack of standardized identity identifiers prevents integration with other platforms or applications.

Wang et al. [11] proposed a method for health data sharing based on a hybrid blockchain forming a consortium chain among health centers and research institutions for secure health data sharing. However, this hybrid chain architecture's construction and maintenance costs are high, and data can only be shared within this consortium, resulting in poor scalability.

Moudoud et al. [12] proposed a blockchain architecture tailored for supply chain applications involving distributed Internet of Things (IoT) entities. The proposed lightweight consensus, LC4IoT, addresses the challenges of high delays and computational requirements. Extensive simulations demonstrate its effectiveness in minimizing computational power, storage, and latency.

The Security Credential Management System (SCMS) [13] provides PKI (Public Key Infrastructure) for vehicular networking, safeguarding privacy through multiple PKI authorities, and Elector-Based Root Management (EBRM). We introduce Blockchain-Based Root Management (BBRM) to enhance decentralization and security, using blockchain to replace the Root Certificate Authority (RCA) and manage elector network membership. Implemented on Hyperledger Fabric, BBRM shows lightweight processing, efficient ledger size, and supports high transaction bandwidth. Our experiments confirm BBRM's suitability for root certificate generation and elector membership control within SCMS. We also analyze how BBRM's parameters impact scheme performance.

In 2021, Fan et al. [14] presented a comprehensive approach to enhancing the security of decentralized applications (dApps) using blockchain technology. The proposed Generic Blockchain Framework (GBF) leverages two blockchains: one to establish trust and another to secure the applications. This dual blockchain setup aims to address fundamental questions regarding blockchain objectives, participants, and consensus protocols. The paper's application of the GBF to various case studies demonstrates its effectiveness and versatility across different sectors, including supply chain,

healthcare, banking, IoT, and networking. While the paper presents preliminary experimental results, the use of two blockchains might introduce additional complexity and computational overhead, potentially affecting the system's efficiency and scalability. Although the GBF is designed to enhance security, the interoperability and synchronization between multiple blockchains could introduce new security vulnerabilities. Furthermore, while the GBF is designed as a general framework, different applications have varied requirements and conditions. Consequently, a highly generalized framework might not optimally meet the needs of specific applications or could perform suboptimally in particular scenarios.

Existing research shows that decentralized technologies are becoming important elements in improving digital identities and data storage. These applications not only enhance data security and privacy protection but also system reliability. However, there is still room for further optimization in terms of costs, identity standardization, and scalability.

3 System Model and Design

This paper proposes a system architecture based on DID to achieve SSI [15,16] and data sharing. Users can create a DID and store their shareable data and identity information in a decentralized system which reduces reliance on centralized organizations or institutions. Users can control access rights to their data. The system architecture consists of three main functions detailed in the subsections.

3.1 Creating DID

Creating a DID is a crucial first step in establishing a user's digital identity for data storage and sharing. Once a user creates a DID, it is registered on the blockchain through a smart contract, ensuring that the identifier can be accessed and verified by anyone through the decentralized network.

The DID creation process, as shown in Fig. 1, involves the following steps:

1. Creates DID: The user initiates the creation of a DID. The system generates a new DID along with a corresponding DID document. This document includes the DID's public key and identity verification methods.
2. Store DID Document on IPFS: The generated DID document is stored on IPFS, a decentralized storage system.
3. Retrieve Hash Value from IPFS: After successfully storing on IPFS, the corresponding hash value for the DID document is retrieved. This hash value is utilized to verify data integrity and locate the data within IPFS.
4. Record Data On-Chain: To conserve transaction costs (gas), only the DID and the hash value of the DID document are recorded on the blockchain. This step ensures that the DID and the hash are permanently recorded in the blockchain to sustain the usability of the identifier and the integrity of the associated identity data.

3.2 Storing Data

Once a user has a DID for identity verification, he/she can store data on the IPFS. The data could be personal medical records, proof of asset ownership, certificates, or any other information the user wishes to share with third parties. After storing the data on IPFS, the corresponding hash value of the data and the user's DID are recorded on the blockchain through a smart contract. This process ensures the integrity and the ownership of the data. The steps for storing data can refer to the process in Fig. 2 and are described as follows:

1. Stores Data on IPFS: The user uploads their data to IPFS.
2. Retrieve Hash Value from IPFS: After successfully stored data, the user can retrieve the hash value corresponding to the data on IPFS. The hash value is used to ensure data integrity and to locate a specific data.
3. Upload Data On-Chain: Using a smart contract, the data's hash value, and the user's DID are uploaded to the blockchain.

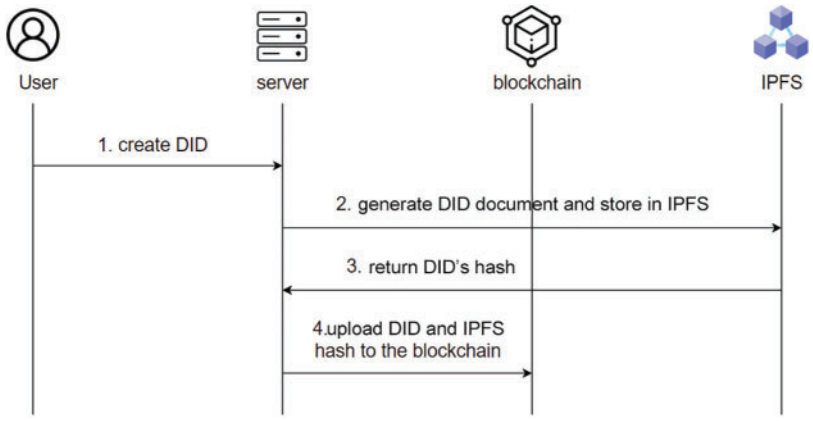

Figure 1: DID creation process

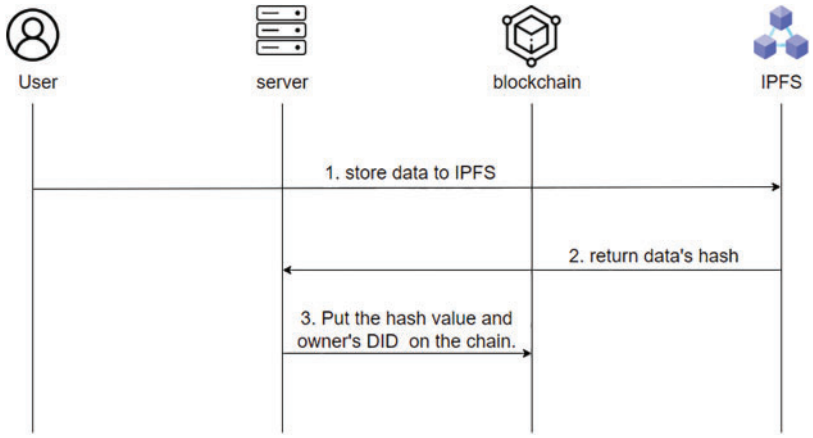

Figure 2: Data storing process

3.3 Sharing Data

The process of data sharing involves three key parts of actions. A data requester first obtains authorization from the data owner to access data stored on IPFS. A Verifiable Credential (VC) is then used to verify the identity of the data requester. If the credentials are validated, the requester is granted with permission to access the data. The data-sharing process is depicted in Fig. 3 and can be outlined as follows:

1. Issue Data Request: A data requester (such as hospitals, research institutions, etc.) submits a data request to the data owner and provides a VC.

2. Retrieve DID Document's Hash Value: Upon receiving the credentials, the data owner retrieves the hash value of the credential issuer's DID document from the blockchain.
3. Retrieve the Public Key: By using the hash value of the DID document, the data owner locates the DID document and obtains the public key of the credential issuer.
4. Validate the Requester's Credentials: The data owner uses the public key of the credential issuer to verify the validity of the requester's credentials. If the credentials are valid, the requester is authorized to access the data.
5. Retrieve Data by Requester: The requester can then retrieve the data from IPFS.

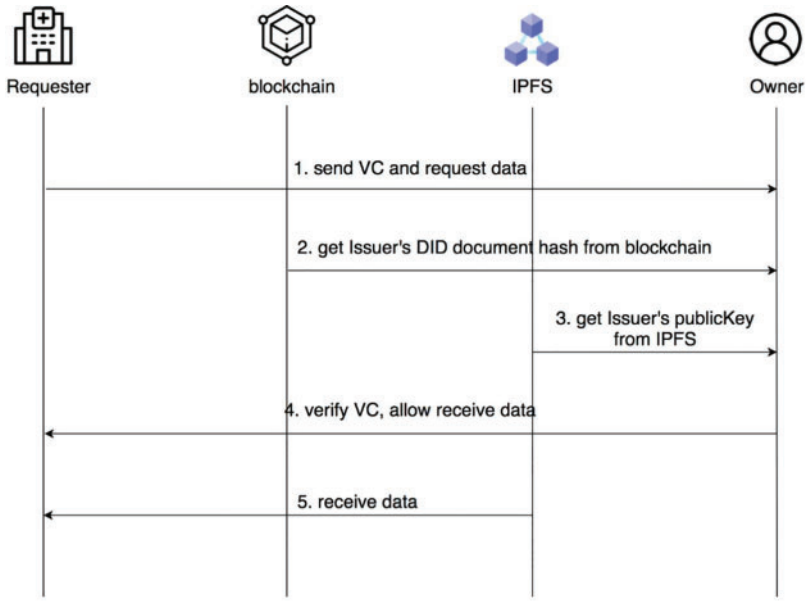

Figure 3: Sharing data sharing process

In the context of data sharing using smart contracts and verifiable credentials, we use two algorithms.

In Algorithm 1, the data requester provides his/her credentials U_{VC}, U_{did}, and the specific data he/she is requesting to the data owner. The data owner uses the issuer's public key $Issuer_{key}$ to verify the validity of the requester's credentials. If the credentials are verified to be valid, the data owner authorizes the requester to access the requested data.

Algorithm 1: Access Authorization

Input: U_{VC}, U_{did}, $DataHash$
1 $signatureHex := U_{VC}.proof.jws$
2 $vp := U_{VC} - U_{VC}.proof$
3 $ddoHash := getDdoHash(U_{VC}.issuer.id)$
4 $Issuer_{key} := getDDO(ddoHash)$
5 $Auth := Issuer_{key}.verify(vp, signatureHex)$
6 **if** $(Auth = True)$:

(Continued)

Algorithm 1 (continued)

$Set\ (U_{did}) \rightarrow Access$
else
return
'Signature is NOT valid. VC verification failed.'

Here is a detailed description of each step in the algorithm:

1. $signatureHex := U_{VC}.proof.jws$

Extract the JSON Web Signature (JWS) from the proof section of the verifiable credential (U_{VC}). This signature is stored in the variable *signatureHex*.

2. $vp := U_{VC} - U_{VC}.proof$

Create a variable *vp* that holds the verifiable credential (U_{VC}) data without its proof section. This step effectively separates the credential's core data from its signature.

3. $ddoHash := getDdoHash(U\_VC.issuer.id)$

Obtain the hash of the DID (Decentralized Identifier) document associated with the issuer of the verifiable credential. The issuer's ID is retrieved from the issuer.id field in U_{VC}, and the hash is generated using the *getDdoHash* function.

4. $Issuer\_key := getDDO(ddoHash)$

Retrieve the issuer's public key using the hash of the DID document obtained in the previous step. The function *getDDO* fetches the DID document based on the provided hash, and the issuer's public key is extracted from this document.

5. $Auth := Issuer\_key.verify(vp,\ signatureHex)$

Verify the extracted verifiable credential data (*vp*) against the provided signature (*signatureHex*) using the issuer's public key. This step ensures that the data has not been tampered with and that it was indeed signed by the issuer.

6. $if\ (Auth == True)$:

If the signature verification is successful (i.e., Auth is True):

7. $Set\ (U_{did}) \rightarrow Access$

Grant access to the DID (U_{did}). This step typically involves setting permissions or providing access rights based on the validated credential.

8. *else:*

If the signature verification fails (i.e., Auth is False):

9. *return 'Signature is NOT valid. VC verification failed.'*

Return a message indicating that the signature is not valid and that the verification of the verifiable credential has failed.

The Algorithm 1 is designed to verify the authenticity of a verifiable credential by (1) Extracting the signature from the credential, (2) Separating the core data from the signature, (3) Retrieving the issuer's public key using the issuer's DID, (4) Verifying the core data against the signature using the public key, and (5) Granting access if the verification is successful, or returning an error message if it fails.

This process ensures that the verifiable credential has been issued by the claimed issuer and that its contents have not been altered.

In Algorithm 2, the smart contract assesses whether the data requester has the necessary permissions to access the data. If the requester is authorized, they can download the data from IPFS using the data's hash *DataHash*. This algorithm provides the processes for issuing credentials and controlling access to a specific data resource (*DataHash*) based on user permissions (U_{did}). Here is the detailed explanation of each step:

Algorithm 2: Issue Credential

Input: U_{did}, *DataHash*

```
for i < AccessList.length do
        if (U_did = AccessList):
                Auth := True
        else
                Auth := False
if (Auth = True):
        allowDownload(DataHash)
else
        return 'You do not have permission to access this file.'
```

1. *for i < AccessList.length do*

Iterate over each item in the *AccessList*. This list presumably contains identifiers of users who have permission to access the resource.

2. *if* ($U_{did} = AccessList[i]$)

Check if the current item in the *AccessList* matches the U_{did} of the user requesting access.

3. *Auth* := *True*

If a match is found, set the Auth variable to True, indicating that the user is authorized to access the resource.

4. *else*

If the current item in the *AccessList* does not match the U_{did}:

5. *Auth* := *False*

Set the *Auth* variable to False. This means that the user is not authorized based on the current item in the list. The loop will continue to check the next item in the *AccessList*.

6. *if* (*Auth* = *True*):

After checking all items in the *AccessList*, if *Auth* is True (indicating that the user's identifier was found in the list):

7. *allowDownload*(*DataHash*)

Allow the user to download the data associated with *DataHash*. This could involve setting appropriate permissions or initiating the download process.

8. *else:*

If Auth is False after checking the entire *AccessList* (indicating that the user's identifier was not found in the list):

9. *return 'You do not have permission to access this file.'*

Return a message indicating that the user does not have the necessary permissions to access the requested file.

The Algorithm 2 is designed to control access to a resource based on a list of authorized users. It works as follows (1) Iterate through the list of authorized users, (2) Check if the requesting user's identifier (U_{did}) matches any entry in the *AccessList*, (3) If a match is found, authorize the user to access the resource, and (4) If no match is found after checking the entire list, deny access and return an error message.

This ensures that only users whose identifiers are present in the *AccessList* can access the specified data, maintaining controlled access to sensitive or restricted resources.

4 System Implementation

This section presents the tools, and function libraries utilized in the implementation process, along with pertinent implementation outcomes.

4.1 Tools and Environment

Table 1 shows the tools and function libraries used in the implementation. Remix is used as the development platform for smart contracts. Remix is an online Integrated Development Environment (IDE) designed for developing Ethereum smart contracts.

Table 1: Tools and environment

Tools	Use
Remix	Smart contract development platform
Ether.js	A JavaScript library for the Ethereum blockchain, used to establish various interactions with smart contracts and the blockchain
Infura	Get a node connected to Ethereum

Remix offers the following functionalities and features that enable developers to easily create, test, and deploy Ethereum smart contracts:

- Online Editor: Remix features an online code editor that allows developers to edit smart contracts directly in the browser. Developers can also use commands to link to local folders for development. This eliminates the need to set up a local development environment, enabling a quick start on smart contract development. By entering the command 'remixd -s [project path] –remix-ide', developers can access local folders through Remix for development.
- Solidity Compiler: Remix uses the Solidity compiler to compile smart contract code and provides the compiled ABI (Application Binary Interface) and Bytecode.
- Integration and Extensibility: Remix can be integrated with other Ethereum tools and services and supports various Ethereum test networks.
- Instant Deployment: Smart contracts can be quickly deployed to Ethereum test networks or the mainnet using Remix. The deployed contract address and transactions can be viewed, and in this research, the contracts are deployed using Remix connected to Sepolia.

Ether.js is a JavaScript library for the Ethereum blockchain, providing developers with the necessary tools and functionalities to facilitate interactions with Ethereum smart contracts and the Ethereum blockchain. Ether.js offers a streamlined and efficient way for developers to integrate blockchain functionality into their projects. As of the latest update, ether.js has reached version 6.9.1. Ether.js is composed of three main modules:

- Provider: This module connects to the Ethereum blockchain. Through the provider, one can query the state of the blockchain, retrieve block information, and access other blockchain-related data. It acts as a gateway to read information from the Ethereum network.
- Signer: Responsible for creating and managing Ethereum transactions and signatures. It signs transactions, enabling users to send Ethereum transactions such as transferring Ether, executing smart contract functions, etc. The Signer module plays a crucial role in the security and execution of transactions on the Ethereum network.

Contract: Used for interacting with Ethereum smart contracts. This module represents a specific smart contract, allowing the invocation of functions defined in the contract and triggering contract events. It serves as an interface for developers to interact with and execute operations defined in smart contracts.

Before interacting with smart contracts, a system needs to connect to an Ethereum network node to execute transactions, query the blockchain state, and interact with smart contracts. Infura is a provider node service that offers Ethereum node services, allowing developers to access the Ethereum network through specific interfaces such as APIs without the need to set up a full node themselves. This service simplifies the process for developers, especially those who may not have the resources or expertise to maintain their own Ethereum node.

4.2 DID Document and Smart Contract

The identity of a DID is reliant on a DID document. By the specifications laid out by the W3C, DID documents associated with a DID are designed. These documents are typically JSON-LD files that define the DID and provide the necessary data for identity verification. This allows other entities to confirm the DID's validity and its owner's identity.

The smart contract plays a crucial role and comprises two main parts:

- Recording and Querying of DID, DID Document, and Data Hash: This part of the smart contract involves uploading the DID, its corresponding document, and the hash value of the stored data to the blockchain. It also includes the functionality for querying these details. This ensures that the data and identity are securely stored and retrievable on the blockchain.
- Control of Data Access: This part manages the permissions for accessing the data. It controls who can access the data based on verifications and criteria defined within the contract.

In the context of data sharing, the thesis employs the W3C's Verifiable Credentials standard. These credentials are formatted as JSON-LD (JavaScript Object Notation for Linked Data) documents and serve to prove certain facts about the DID that possesses the credential. This verifiable certificate can prove that the did is a researcher of a certain research lab. The data owner verifies the validity of this certificate through the issuer's public key, and if the certificate is valid, the data owner authorizes the requester to have access to the data.

5 System Analysis

5.1 Comparative Analysis with Other Related Literatures

As shown in Table 2, our proposed method is compared with related literature in various aspects, such as technology, digital identity category, application range, scalability, development cost, storage cost, integrity, and the risk of loss.

Table 2: Comparative analysis

Methos	Naik et al. [7]	Bhandari et al. [9]	Wang et al. [11]	Our method
Technology	Ethereum blockchain	IPFS with Ethereum blockchain	XuperChain	IPFS with Ethereum blockchain
Identity	Decentralized identity	Centralized identity	Federated identity	Decentralized identity
Application range	Health records	Health records	Health data sharing	Various data types including health records, financial data, IoT data, and more
Scalability	Medium	Medium	Low	High
Development cost	Medium	Medium	High	Medium
Storage cost	High	Medium	Medium	Medium
Integrity	High	High	High	High
The risk of data loss	Low	Low	Medium	Low

Different from other methods, our method adopts IPFS with Ethereum blockchain and decentralized identity in terms of technology and identity. In terms of application range, Naik et al. [7] focused on managing and storing health records, ensuring patient data is secure and easily accessible within the healthcare industry. Bhandari et al. [9] concentrated on health records with a centralized identity system, suitable for institutions where centralized control and quick access are priorities. Wang et al. [11] emphasized health data sharing among various entities, facilitating data exchange and collaboration within healthcare networks. Our method designed for a broad range of applications, including health records, financial data, IoT data, and more. This versatility makes it suitable for multiple industries requiring secure, scalable, and cost-effective data management solutions.

Scalability stands as a pivotal consideration when selecting a DID (Decentralized Identity) system. Decentralization offers unparalleled scalability and flexibility, whereas consortium chains lag in both aspects compared to public chains, rendering them less scalable.

Concerning development costs, utilizing a consortium chain proves more expensive, necessitating a more intricate setup, management, and maintenance. Consequently, the third option is costlier to develop. Conversely, leveraging a public chain is more cost-effective since there is no need to construct and maintain an underlying blockchain network.

In terms of storage costs, blockchain storage incurs higher expenses due to transaction fees. Hence, the first option is pricier to store. Cloud server storage costs less and varies mainly based on service providers and usage volumes. IPFS storage boasts the lowest costs and mitigates reliance on centralized systems.

Blockchain's immutability ensures data integrity. Finally, regarding the risk of data loss, consortium blockchain systems typically store data in a limited number of nodes. While this boosts efficiency and privacy, it raises the risk of data loss if nodes fail or fall victim to attacks. Therefore, the third method carries a higher risk of data loss.

The proposed system architecture stores data on IPFS, uploading only hashes to the chain to maintain data immutability while circumventing the costly transaction fees of uploading all data to the chain. Moreover, employing standard-compliant decentralized identities and implementing them on the widely used Ethereum chain markedly enhances digital identity scalability. Public chains also entail lower development costs than consortium chains. With our average upload time to IPFS standing at 24 s per transaction, and considering the average transaction fee of 0.003788 Eth, this information can provide valuable reference for future planning and scheduling work.

Additionally, exploring external storage mechanisms for IPFS beyond the blockchain can provide insights into alternative approaches for decentralized data storage and retrieval. The InterPlanetary File System (IPFS) is a peer-to-peer protocol designed to facilitate decentralized file storage and sharing. While IPFS itself is not a blockchain, it can be used in conjunction with blockchain technology to enhance data storage and retrieval. IPFS stores data in a distributed manner across a network of nodes, offering benefits such as content addressing, deduplication, and resilience against censorship.

External storage mechanisms for IPFS apart from the blockchain do exist. For example, IPFS can be integrated with traditional cloud storage solutions, such as Amazon S3 or Google Cloud Storage, to provide decentralized access to stored data. Additionally, IPFS can be used in conjunction with distributed storage networks like Filecoin, which incentivize users to contribute storage space in exchange for cryptocurrency rewards. These external storage options offer flexibility and scalability for IPFS-based applications, enabling developers to tailor their storage solutions to specific requirements and constraints.

5.2 Security and Availability Analysis

The implementation of blockchain technology, distributed storage, and self-sovereign identity (SSI) management in this paper enhances security, privacy, and availability:

- Security and Privacy Self-Sovereignty: Decentralized identity allows users to own and control their personal identity data, reducing reliance on centralized organizations. It minimizes the risk of data breaches and misuse.
- Verifiability and Transparency: DID, once registered on the blockchain, is accessible and verifiable by anyone. The immutability of blockchain guarantees the integrity and accuracy of the data.
- Scalability and Compatibility: The DID design is based on W3C standards to ensure interoperability and cross-system/cross-platform identity verification. Users can use the same identity across multiple platforms and services without the necessity to create and manage individual identity separately which enhances user convenience.
- Persistence and Availability: Blockchain-based identities are persistent and unalterable once they are recorded to ensure long-term reliability. The decentralized nature of IPFS avoids single

points of failure, thus enhancing data availability. Even if some nodes fail, the data remains accessible and ensure consistent service.

6 Conclusion

In this paper, we propose a framework using SSI and data sharing based on DID and VC. This architecture allows users to create their own identity identifiers to store and control their sharable data, thereby enhancing users' privacy and their controls over data access rights.

Utilizing internationally standardized decentralized identifiers enables the de-identification of identity data and increases the scalability of the system. This decentralized identity recognition hands back the control of the personal identity to users themselves, allowing them to freely authenticate and exchange data with other entities while protecting their privacy and data security.

Based on the characteristics of blockchain technology, such as immutability, decentralization, traceability, and transparency, our framework effectively enhances the scalability of the system and the integrity of the data. The system stores DID documents and data to be shared on IPFS by uploading only the hash and corresponding DID to the blockchain. This approach significantly reduces storage costs while ensuring the availability, validity, and accuracy of the data.

Our proposed framework offers greater scalability, usability, and security than other related literature. The application range of DID is extensive as well. The DID framework is not only applicable to data sharing as described in the paper but also can be further extended to other applications in Web3 in the future.

Our experimentation with the Ethereum public blockchain highlighted challenges such as slow transaction speeds and high fees. Despite these limitations, our proposed framework's significance remains undiminished. While still in its academic research phase, we are optimistic about the potential for more efficient public blockchains to support our mechanism in the future. We appreciate feedback and are committed to refining our research to address these concerns.

Acknowledgement: The authors would like to express their gratitude to the members of the research group for their support.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Study conception and design: Iuon-Chang Lin, I-Ling Yeh; simulation, analysis and interpretation of results: Iuon-Chang Lin, I-Ling Yeh; draft manuscript preparation: Iuon-Chang Lin, I-Ling Yeh, Ching-Chun Chang, Jui-Chuan Liu and Chin-Chen Chang. All authors reviewed the results and approved the final version of the manuscript.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Federal Trade Commission. FTC imposes $5 billion penalty and sweeping new privacy restrictions on facebook. 2019. Available from: https://www.ftc.gov/news-events/news/press-releases/2019/07/ftc-imposes-5-billion-penalty-sweeping-new-privacy-restrictions-facebook [Accessed 2024].
2. The New York Times. FTC investigating twitter for potential privacy violations. 2020. Available from: https://www.nytimes.com/2020/08/03/technology/ftc-twitter-privacy-violations.html [Accessed 2024].

3. Sporny M, Longley D, Chadwick D, Steele O. Verifiable credentials data model v2.0. W3C. 2024. Available from: https://www.w3.org/TR/vc-data-model-2.0/ [Accessed 2024].
4. Sporny M, Longley D, Reed D, Steele O, Allen C. Decentralized identifiers (DIDs) v1.0. W3C. 2022. Available from: https://www.w3.org/TR/did-core/ [Accessed 2024].
5. Zyskind G, Nathan O, Pentland AS. Decentralizing privacy: using blockchain to protect personal data. In: Proceedings of the 2015 IEEE Security and Privacy Workshops, 2015 May 21–22; San Jose, CA, USA.
6. Benet JB. IPFS-content addressed, versioned, P2P file system. Netw Internet Archit. 2014 Jul. doi:10.48550/arXiv.1407.3561.
7. Naik KV, Murari TV, Manoj T. A blockchain based patient consent management technique for electronic health record sharing. In: Proceedings of the IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2022 Dec. 1–3; Mangalore, India; p. 1–3.
8. Antonopoulos AM, Wood G. Mastering Ethereum: Building smart contracts and dapps. Sebastopol, CA, USA: O'Reilly Media; 2018.
9. Bhandari B, Vairagade R, Trivedi H, Thakre H, Indurkar G, Yadav A. Decentralized medical healthcare record management system using blockchain. In: Proceedings of the 11th International Conference on Emerging Trends in Engineering & Technology—Signal and Information Processing (ICETET-SIP), 2023 Jun; Rajkot, India; p. 1–5.
10. Otta SP, Panda S, Hota C. Identity management with blockchain: indian migrant workers prospective. In: Proceedings of the IEEE Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), 2023 Mar; Bhubaneswar, India; p. 1–6.
11. Wang T, Wu Q, Chen J, Chen F, Xie D, Shen H. Health data security sharing method based on hybrid blockchain. Future Gener Comput Syst. 2024 Apr;153(2):251–61. doi:10.1016/j.future.2023.11.032.
12. Moudoud H, Cherkaoui S, Khoukhi L. An IoT blockchain architecture using oracles and smart contracts: the use-case of a food supply chain. In: Proceedings of the IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2019 Sep; Istanbul, Turkey; p. 1–6.
13. Sarker A, Byun SH, Fan W, Chang SY. Blockchain-based root of trust management in security credential management system for vehicular communications. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing, 2021 Mar. 22–26; Gwangju, Republic of Korea; p. 22–6.
14. Fan W, Hong H, Zhou X, Chang S. A generic blockchain framework to secure decentralized applications. In: Proceedings of the ICC, 2021–IEEE International Conference on Communications, 2021 Jun; Montreal, QC, Canada; p. 1–7.
15. Mühle A, Grüner A, Gayvoronskaya T. A survey on essential components of a self-sovereign identity. Comput Sci Rev. 2018;30:80–6. doi:10.1016/j.cosrev.2018.10.002.
16. Ferdous MS, Chowdhury F, Alassafi M. In search of self-sovereign identity leveraging blockchain technology. IEEE Access. 2019 Jul;7:103059–79. doi:10.1109/ACCESS.2019.2931173.