Identity ledger on Blockchain

Final Report

Ayush Sharma
University of Florida
UFID: 4034-1961

Subham Agarwal
University of Florida
UFID: 7949-7379

Saheel Ravindra Sawant
University of Florida
UFID: 1164-7923

Rohan Sanjay Shahane
University of Florida
UFID: 6859-1943

ABSTRACT:

The rise in the use of the internet allows us to take a step towards the paperless and digital identification of individuals. The physical/paper documents which were previously used by individuals to prove their identity can now be replaced with their digital counterparts thanks to the internet and cloud services. Although it provides us with additional convenience and ease of use, it also introduces privacy risks associated with the malicious and unintentional use of the documents out on the web. In order to limit the distribution of an individual's personal documents to unintentional persons/entities, we are proposing a blockchain-based solution which will give end-users fine-grained control over who has access over their personal documents. We demonstrate this with a web-based Dapp, where the user can specify the individual/organization to whom he wants to send its document.

Index Terms - Blockchain, IPFS, Smart contracts, Ethereum, Ganache.

I. INTRODUCTION

In recent decades, people from multiple demographics have easy access to advanced technologies like mobile phones, personal computers, laptops, smartwatches, and many other electronic devices. Among the various applications available, social networking portals have become an integral part of people's life. It generates and processes tons of data every day. This data also contains sensitive user information like gender, age, banking credentials, or forensics like fingerprints, retinal scans, etc. that are stored by these applications. The combination of these and other data values constitute a virtual identity of the user. Often, this information associated with the user gets transferred across different networks, and thereby it is accessible to users or organizations that do not have proper privileges to access, use, or modify this data. Such a breach of private information might directly or indirectly cause some financial losses and further, destroy the trust of users as well as the sanctity online transaction.

According to the US consumer data breach report [6] of 2019, 2.8 billion consumer data records were exposed to cyber attackers in the year 2018, at an estimated cost of $654 billion. 34.2% of those cyberattacks were classified as unauthorized access. Due to such incidences, it underlines the need that an individual should have more fine-grained control over their personal data and who has access to it. Along with managing who has access to the user's personal data, an individual should also know who is using their personal data and for what purpose. This helps to ensure the privacy and security of the user data is preserved.

Even in the real world, several instances, like visiting a bank, registering for network services, claiming healthcare or insurance benefits, going to the electoral voting, etc., require the users to show their official identification documents like social security card, driver license, photo ID to verify their individual identity. People need to carry their documents physically and sometimes are asked to submit copies of these said documents. Once the copies are submitted for further verification, there's always a possibility that they might get into the wrong hands and the sensitive information mentioned therein, being used for unethical and unlawful practices.

In order to address this problem of access control with respect to the user information, we propose a decentralized identity ledger based on blockchain. The reason behind using blockchain to implement this functionality is the wide range of features that it offers. Like the decentralization of the database ensures, data is divided and spread across multiple nodes and there's no single source of failure. Also, the requirement of consensus among all the nodes or the majority of nodes on the network, makes it a lengthy and

complex process for malicious nodes to alter the rules concerning data access. The scalability of blockchain allows new users to join the chain and generate their identity and create access rules for their private data. The openness of the blockchain makes it explicit for all the information to be hashed before it is stored on the chain. Due to the encryption of data, the actual information remains secure and stays hidden from the other nodes.

Our proposed system aims to give end-users the ability to own and have fine control over their personal data. The access control mechanism allows the end-user to grant, modify, and revoke access to its personal documents. To demonstrate our idea, we have created a website on the localhost and have linked it to the Ethereum platform. To handle the data files, the Interplanetary File system (IPFS) API has been used and to access the Ethereum enabled distributed applications, the metamask extension is added. This paper is structured as follows: Section II contains an overview of the previous research in context with access management for user data. Section III explains the system design of the system. Section IV constitutes the implementation details and the solidity code. Section V addresses the scalability of the system in terms of varying data sizes and the number of users. Section VI provides an analysis concerning the security of the system. Lastly, in section VII, we provide a conclusion and discuss the future scope.

II. LITERATURE REVIEW

In paper [7], Adya Kiran et al. describe how the blockchain could be used as an underlying framework to provide access to private user information retrieved through an application or website. They put forth an approach that can reduce the reliance on a third-party organization and decentralize the authorization process. Between the resource and the node requesting the transaction, the chain acts as an abstract layer, and all the requesting parties are identified by a unique identification number. Further, to enforce the terms of access on the user data, smart contracts can be executed. They ensure the requester can access only those attributes of the data, which the owner has pre-approved. These smart contract codes are immutable, self-executable, provide open verification, and certain cost reductions. In this architecture, the owner of the data deploys the policy definition contract and it stores the data access privileges. On the other hand, the requester can invoke the data access contract and is granted certain attributes if that node has permission for the same. This model is similar to that of an Attribute-Based Access Control (ABAC) model, but the important distinction is that the attributes considered are that of the user's data and not of the user. The authors also address the privacy concerns associated with real-time implementation and provide a few probable solutions like encryption of the pointer to the data source or the queries that are made to the data source.

In paper [8], Yuan Liu et al. suggest a decentralized system for managing identity, which can be implemented using blockchain. This model focuses on two key concepts, namely identity authentication, and reputation management. For all the user nodes involved, the information regarding their personal identities, and the reputation will be stored in the blocks. As a reflection of a node's reputation, a token (RpCoin) is generated upon executing the smart contract. This token is similar to a digital certificate representing the user's reputation. It cannot be used or transferred to other nodes. If any node participates in a reputation or incentive task, then the number of RpCoins for that node gets increased whereas a negative participant is punished with a reduction in the number of RpCoins. The reputation tasks consist of two stages: the initial publishing stage and the consensus stage. In an incentive task, a node is given bonus RpCoins if it points out the negative workers on the network or the nodes which are repeatedly circulating previously approved proof of works (PoW). The proposed system model prevents the creation of several identities for one physical entity, thereby extending the potential to address the Sybil attack[9]. The physical entity can create only one virtual identity taking advantage of the immutability feature of the blockchain. Also, the identity modification procedure allows honest nodes to avoid the whitewashing attack[10].

In [11], Rouhani and Deters, discuss the different blockchain-based access control applications implemented till date and its challenges that need to be addressed. Like, the attribute-based encryption methods [12] that are susceptible to privacy leakage from the private key generator as well as a single point of failure. Also, the management of such methods is difficult. Here, Enigma [13] framework is discussed, which allows its users to have control over their data and alter or rescind access to the data in a transparent way. It consists of three distributed databases, namely a blockchain, a distributed hash table, and a multi-party computation database that breaks down data and allocates without replication to the different

nodes on the network. Further, the access control mechanism is explained in terms of cloud federations, multiple organizations, shared blockchains, and self-sovereign identities.

In [14], Zhang et al. describe how smart-contract based models could be implemented to regulate access control for the Internet of Things (IoT). The access control contracts consist of a 'judge contract' and a 'register contract'. The judge contract uses a method to determine any misbehavior by other nodes on the chain and decides a penalty accordingly. The register contract methods fetch and provide information regarding access control and help to register or update permissions. For each subject-object pair, there's a method that performs both static and dynamic right validation based on the behavior of the node. To demonstrate the validity and feasibility of this framework, an IoT system with multiple large servers, storage devices, IoT gateways, etc. connected through a peer-to-peer network is used.

Out of the traditional access control models like role-based (RBAC), the attribute-based (ABAC), or the capability-based (CapBAC), Jason Paul Cruz et al. in [15], base their research on the role-based access control model. Such a framework can be used for trans-organizational communication. The RBAC, along with blockchain and smart contracts, can be realized as a secure, user-oriented, verifiable model. The owner node could issue a challenge-response protocol to confirm the role of the recipient node. This protocol doesn't require the two involved parties to form a partnership or interact with one another. It can be computed online as well as offline. Such a model doesn't allow other entities to perform specific actions without authorization, however, it is highly dependent on the role-issuing organization which has the majority of the control. Such dependence could be a single target for failure. Also, it works for a predefined set of nodes and requires a lengthy procedure when a new organization or an unfamiliar node joins the network.

III. SYSTEM DESIGN

A. BLOCKCHAIN

Blockchain is the backbone of our proposed system. As the name suggests, it is a chain of 'blocks' that contains the transaction data, a cryptographic hash of the previous block, and a timestamp. The blockchain can be described as "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way"[16]. The first block of the chain is known as a genesis block. It is the starting point of the blockchain and a special block in the blockchain which is not linked to any other previous blocks[17].

A block in the blockchain is linked with the next block by storing the cryptographic hash value of its previous block in the current block [20]. The cryptographic hash of a block is based on the data which is contained in that block, which means if the contents of a block have been tampered with even slightly, the cryptographic hash associated with that block will change and the chain will be invalidated. To change a previous block, an attacker must have to redo the proof-of-work of that particular block and all blocks after it. Also, it needs to catch up with the progress of the honest nodes on the network. [3] [18]

Blockchain has 'miners' who validate the transactions from the transaction pool and then append to the ordinal blockchain. The way miners carry this task out in the bitcoin network is by first selecting transactions from the available transaction pool. Miners get this block into a candidate block, now miners try to add this candidate block to the blockchain. Miners compete with each other to add the block into the blockchain. The block that is to be added into the blockchain is decided by calculating the hash of the block and if that hash is below a certain target value, then that block will be added to the blockchain. The difficulty to get this target value determines the time it will take to mine a block. The bitcoin network averages around 10 minutes for one block[21].

The most popular implementation of blockchain-Bitcoin cryptocurrency[3], stores transactions of bitcoin tokens among users on a blockchain. This ensures the authenticity of the transaction and secures it against modification. Similarly, we propose to store the encrypted IPFS hash of the identification documents, as a transaction on the blockchain. The blockchain stores the recipient's account address and the encrypted hashcode as a key-value pair.

The property of the blockchain ensures that this key-value pair once written on the blockchain can not be modified or tampered with, this entry in the blockchain keeps the record of who has access to a particular document uploaded by the user. Upon request by a beneficiary, the

system checks if the requester has access rights to get this document by searching the blockchain with the requester's address. If a corresponding entry is found in the blockchain for the provided address, the respective hashcode of the file is returned back to the requesting user.

B. Interplanetary File system

Our application proposes a secure mechanism to store and share a user's identity data by storing it on a blockchain. However, storing any significant amount of data on a blockchain is extremely expensive [1], as it will require a huge amount of computational power to write the large blocks. One solution to this predicament is to store the actual file on a third party cloud server and then store only the access information on the blockchain. Following this approach does solve the price issue but it also introduces a central authority having control over user's data, which goes against the core principle of decentralization of blockchain. Therefore in our application, we store the data on the Interplanetary File System or IPFS.

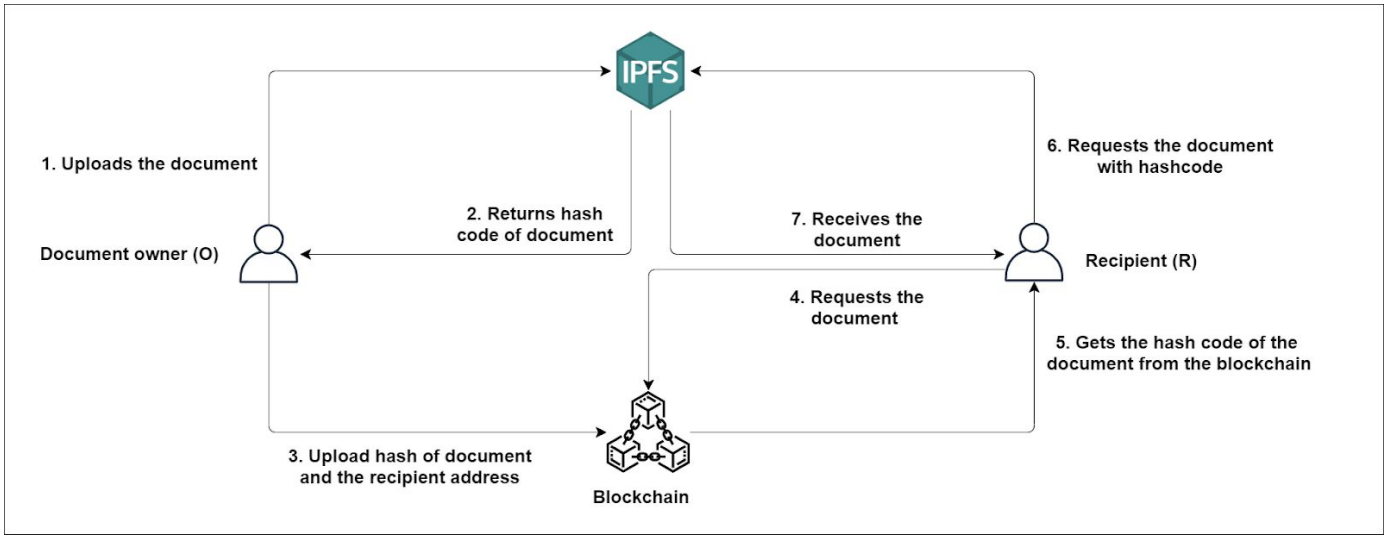

Fig. 01: System Architecture

IPFS is a peer to peer network for storing and sharing data in a distributed file system [2]. We choose IPFS because it is a distributed file system and is not owned by a central authority. IPFS divides files into chunks of smaller sizes and stores them on multiple nodes of a distributed network. Each node on the network is responsible for a set of chunks. All the chunks are hashed together to form a Merkel tree. Each leaf of the Merkel tree is one constituent chunk of the file. These are then pairwise merged to form a root node. The root of the Merkel tree is hashed and its hashed string is returned as the address of the uploaded file. When a file is requested using the root node of this tree, all its constituent chunks are searched, combined, and sent to the user. In this way, IPFS achieves content addressing which means it uniquely identifies any data on the file system by a fixed hashed value rather than its physical location on the network.

This hash value used in forming the tree is obtained by running a hashing function on the uploaded documents. The hashing functions used in this system are deterministic hashing functions. This means that for a given input value the function will always produce the same output string. But it is really difficult to guess the input data by observing the hash string. This property plays a significant role in our system as these strings can be used to uniquely identify the contents of a file. This is useful in implementing a verification mechanism on the application. Any authentic document should produce the same string every time it is hashed using a deterministic hashing function. Any document uploaded to IPFS is hashed to form a string of fixed length that is independent of the size or type of data. The contents of the data can not be determined just by observing the hashcode value. This makes it a suitable choice to identify a user's data on the network. Our application then stores this hashcode in the form of a string on the blockchain.

IV. IMPLEMENTATION

The system implementation is based on the Ethereum platform [4]. Ethereum is an open-source, blockchain-based, decentralized software development platform used for creating smart contracts or decentralized applications. Ethereum is a public blockchain which can be joined by any user. Each user in the network acts as an Ethereum node and consensus between these nodes is used to agree upon a transaction made in the network. Ethereum uses its own cryptocurrency called ether to make payments to these nodes for sharing their computing power. We used truffle to develop, deploy, and test smart contracts.

the encrypted hashcode of the file stored in IPFS as a key-value pair. We utilized the ipfs-API module in order to store the file in the Interplanetary File on a remote IPFS node (provided by INFURA for developers) at the address ipfs.infura.io:5001 and generate an IPFS address hash. This address is further encrypted using the eth-crypto module and the recipient's public key so that only the intended recipient can decrypt the message. We also provided a decryptMessage module that takes the encrypted String and the recipient's private key as input. This can be used to retrieve the IPFS address hash which gives us the shared file.

In the IdentityLedger contract, the function sendIPFS is

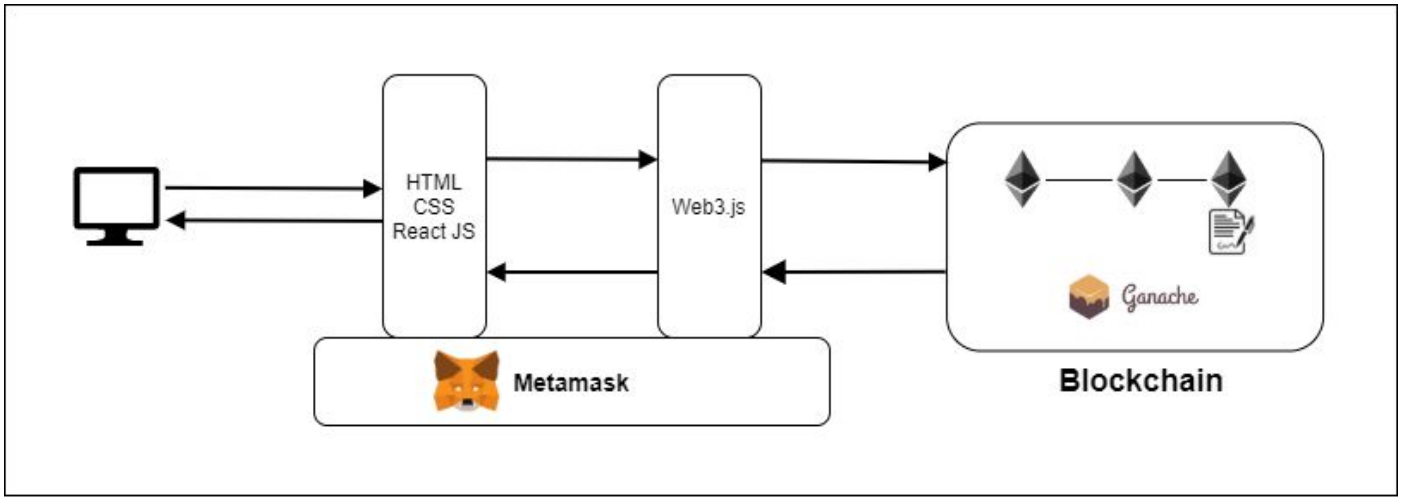

Fig. 02: Application components

A smart contract [5] is a small computer program that is used to verify and enforce a contract between two negotiating parties. We are using the smart contract to store the above mentioned key-value pair. This smart contract is deployed on the Ethereum network. The Ethereum network charges a small fee for deploying a smart contract that can be paid with ether. Therefore, a user must have ether to develop applications and deploy smart contracts on the Ethereum network. This can be problematic for developers who want to develop and test distributed applications. One solution to this problem is to use a personal blockchain to run tests. We have used ganache as the personal blockchain to deploy contracts and run tests on our application.

We used solidity as the programming language to implement the IdentityLedger contract and deployed it in our private blockchain. It stores the recipient address along with used to store the recipient's address along with the file's IPFS address hash. Whereas, the function checkInbox is used to check if there's any record stored for him in the contract. The checkInbox function not only checks if the user calling the function has an IPFS address hash associated with its address, but it also uses the property msg.sender to do so ensuring that the message to the contract was signed by the calling entity.

As it is not possible to return values from functions which modify the state of the blockchain we used solidity events in order to return information in two scenarios -

1. when the hashcode is stored on the blockchain, ipfsStored, and
2. when the recipient tries to check if there's any associated hashcode stored for his address, receivedIPFS.

These events are also invoked inside the sendIPFS and checkInbox functions.

We also provide a web user interface in order for the clients to be able to access and utilize this smart contract. So we bootstrapped our project with React Truffle Box and integrated the smart contract to build a decentralized application. MetaMask is used in order to make Web3.js API available for the application to communicate and interact with the smart contract. MetaMask also provides a secure interface to review and sign the blockchain transactions. The component componentDidMount initializes web3 which allows us to interact with our local ethereum node, using an HTTP connection. The component also synchronizes the available accounts, loads the IdentityLedger contract from truffle, and returns an instance of the deployed contract. The instance is further used to check if there are existing events for a particular account. Finally, to simulate a local and private blockchain - we used Ganache to run our test cases and inspect the state of your Dapp.

The following table shows the gas used and the transaction fee for performing the following tasks:

Function	Gas used	Cost
Initial Migration	196887	0.00393774 ETH
Deploy Contracts	472443	0.00944886 ETH
Share a file	270774	0.00541548 ETH
Download a file	42979	0.00085958 ETH

Here, we provide the solidity code of our smart contract:

```
pragma solidity ^0.4.23;
contract IdentityLedger {

        mapping(address=>string) identityLedger;
        event ipfsStored(string _ipfsHash,
        address _address);
        event receivedIPFS(string response);
        modifier notFull(string _string) {
                bytes memory stringTest =
                bytes(_string);
                require(stringTest.length == 0);
                _;
        }
        constructor() public {}

        function sendIPFS(address _address,
        string _ipfsHash) public
        notFull(identityLedger[_address]){
                identityLedger[_address] =
                _ipfsHash;
                emit ipfsStored(_ipfsHash,
                _address);
        }

        function checkInbox() public {
                string memory ipfs_hash =
                identityLedger[msg.sender];
                if (bytes(ipfs_hash).length == 0)
                {
                        emit receivedIPFS("Empty
                        Inbox");
                } else {
                        identityLedger[msg.sender]
                        = "";
                        emit
                        receivedIPFS(ipfs_hash);
                }
        }
}
```

\client\src\encrypt.js

```
const EthCrypto = require("eth-crypto");
const encryptMessage = async (pubKey, message)
=> {
        pubKey = pubKey.substring(2);
        const encrypted = await
        EthCrypto.encryptWithPublicKey(
                        pubKey,
                        JSON.stringify(message)
        );
        return
        EthCrypto.cipher.stringify(encrypted);
};

const decryptMessage = async (privKey,
encryptedString) => {
        const encryptedObject =
        EthCrypto.cipher.parse(encryptedString);
        const decrypted = await
        EthCrypto.decryptWithPrivateKey(
                privKey,
                encryptedObject
        );
```

```
        return decrypted;
};
export { encryptMessage, decryptMessage };
```

\client\src\ipfs.js

```
const IPFS = require("ipfs-api");
const ipfs = new IPFS({
        host: "ipfs.infura.io",
        port: 5001,
        protocol: "https"
});
export default ipfs;
```

\client\src\app.js

```
componentDidMount = async () => {
        try {
                const web3 = await getWeb3();
                const accounts = await
                web3.eth.getAccounts();
                const Contract =
                truffleContract(IdentityLedger);
                Contract.setProvider(web3.current
                Provider);
                const instance = await
                Contract.deployed();
                instance.receivedIPFS().on("data"
                , result => {
                        this.setState({
                        receivedIPFS:
                        result.args[0] });
                });
                this.setState({ web3, accounts,
                contract: instance });
        } catch (error) {
                alert(`Unable to load web3,
                accounts, or contract.`);
                console.error(error);
        }
};
```

V. SCALABILITY

In this section, we postulate how our system will perform when it experiences a rise in the number of users. We analyze our system for two scenarios, first when the total number of users of the system increases and second when the size of the data being shared by an individual user increases.

A. Varying Data Sizes

We attempt to examine the system for various sizes of files uploaded onto the system. This helps us analyze the system's response when a file of large size is shared by an owner. We inspect the cost of sharing large files on the system. It is observed that there is no effect of the size of the file on the cost of storing it on the Ethereum.

Irrespective of the size of the file being stored, the interplanetary file system used in this application returns a string of fixed length. This string generated is the hash value for the file. The size of the address string does not depend on the size of the file stored. Therefore the size of data being saved on the Ethereum blockchain remains constant even if there is an increase or decrease in the size of the file. This means that there is no additional cost of saving larger file sizes on the system.

From the performance perspective, IPFS has no limits on the size of a single file that can be stored. Although the Infura API that we have used in our implementation has an upper limit of 100Mb for a single file. Larger files are divided into smaller chunks and stored on different nodes. With an increase in the size of the file and the number of chunks created, more time may be spent in searching for the chunks while retrieving the file. Although there is no significant latency in data retrieval for significantly large file sizes in IPFS. Moreover, any latency in file retrieval in IPFS can be the result of the underlying network latency instead of the actual file size.

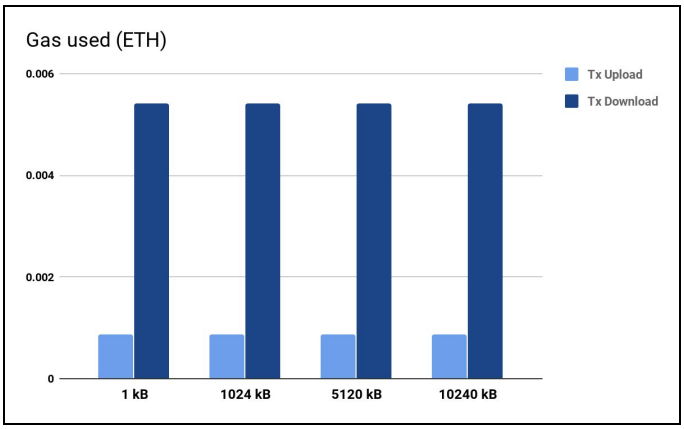

Fig. 03: Gas used for varying file sizes

B. Varying the Number of Users

In the current implementation of the application one transaction is created every time an owner wishes to share his data with a recipient. This means that even if an owner shares only 1 file with 5 different recipients, 5 transactions are created in the blockchain. This incurs additional costs for the file owner. As a result, the number of transactions created will rise exponentially with an increase in the number of users on the system. Assuming every user is sharing files with multiple users. The increase in the number of transactions also results in a performance loss as each transaction needs to be approved on the chain. As the throughput of ethereum blockchain is limited this will cause delays in the transaction being approved and the document is made available to view.

These additional costs and performance lag can be prevented by introducing the concept of compound keys. As described in [19], a compound identity can be created for documents that need to be stored over the system. A compound identity is a common identity shared by multiple users, one of which is the owner of the identity. A compound identity is comprised of five entities, namely:

(K_{pub}^{O}) - publicKey of owner
(K_{pri}^{O}) - private key of the owner
(K_{pub}^{R}) - public key of the recipient
(K_{pri}^{R}) - private key of the recipient
($K_{Sym}^{O,R}$) - symmetric key

Every time an owner wishes to share a new document, it can generate a symmetric key and encrypt the file's hash code with that symmetric key before storing it on the blockchain. Now every time the file needs to be shared with a new recipient he can be included in the compound identity. The owner will just open a connection with the recipient, send him his public key, and the previously generated symmetric key for this identity. The recipient can then use this symmetric key to decrypt the received file's hash. This will reduce the number of transactions on the blockchain as only one transaction will be required per document.

VI. SECURITY ANALYSIS

In this section, we evaluate the security of the proposed system. We achieve this by evaluating our system against various possible threat models and propose mechanisms by which the system promises better security. The system aims to secure a user's personal data against unauthorized access and accidental loss. We have made certain assumptions while evaluating and implementing the system.

Assumption 1: We assume that the owner of the documents and the intended recipient for the documents have a channel for communication off-chain which they can also use to share the recipient's public key once.

Assumption 2: We assume that only the authentic user is in possession of the respective private key of the user. And it is the user's responsibility to store the private key at a secure location.

Now we discuss possible threat models to the system and how the system handles each one of them.

Threat model: An adversary gains access to the hashed string that is used to request files from the IPFS.

Argument: A non-user adversary may snoop in on the communication between the system and the recipient and receive the hash string that is stored on the chain. But he can not decrypt it to find the string that can be used to request the file on the interplanetary file system He will need the authentic recipient's private key to decrypt the string and this prevents the file from unauthorized access by the adversary.

Threat model: A user who is not intended to have access to a particular file may attempt to receive the file. A registered user can easily read the public ledger and read the account ID of an authorized user and may request the system for the files by posing as an authorized user.

Argument: A user might read the account ID that has authorized access to a file from the public ledger. Using this he can pretend to be the authorized user and gain access to the files hash code. But the user can not decrypt this hash code to get the content address of the file on the file system. This prevents the user from gaining access to the file.

Threat model: A malicious user can attempt a denial of service attack on the system and try to prevent a genuine user from receiving the file. It might be in the user's best interest to delay the service.

Argument: A malicious user might attempt to overburden the system by sending a large number of bogus file requests to the file storage. But our design decision to use IPFS instead of a traditional server makes it highly unlikely for the malicious user to be successful in denying a genuine user to get access. Because of IPFS's distributed design and abstraction about the actual location of a file, it is highly unlikely that an attacker can attack the location of a file.

Threat Model: A malicious recipient may try to misuse a file once he has received it from the system.

Argument: The file owner node (O) may grant access to his identity documents to a user A through the system. After the user A has verified his identity and downloaded the file from the system, he can use the file for his off-network use. Suppose the user sends the file to user B. For the identity document to be of any benefit to anyone, it must be used as an identity proof at a genuine platform. Even if the platform is off-system, the receiver can verify the authenticity of the document by querying the system about the ownership of the document. In a future implementation, a verification mechanism can be added to the system which will check if the origin of a file is through the chain. In the given example if node A pretends to be the actual owner of the document, then there must be an entry on the blockchain with his signature granting access to an authorized user. But when node B attempts to verify this on the chain, he finds out the only entry for this document on the block is that of a node O sharing it with node A which places node O as the owner of the document. In this case, the system can warn the user of possible malicious use of the documents.

VII. CONCLUSION AND FUTURE SCOPE

In this paper, we have proposed a system that ensures the secure management of a user's data over the blockchain. It successfully addresses the issue of data loss while sharing physical copies of documents. It also aims to provide the user with more transparency and nuanced control over his data. Additionally, we have analyzed the system for vulnerabilities based on various threat models.

The implementation we have provided with this paper handles a user's identity documents. But with further advancement in technology and its ubiquitous presence, a person's identity in the future will live in digital forms. Future iterations of the application can aim to incorporate files that encode a user's digital signatures like fingerprints and iris scans. The effects of scaling the users on the system remains a topic of further study. Furthermore, the implementation of mechanisms to include the user's digital signature with the document can improve the authenticity of data on the application. This will ensure that the documents received off the application can also be verified for authenticity.

REFERENCES:

[1] Matzutt R, Hohlfeld O, Henze M, et al. POSTER: I Don't Want That Content! On the Risks of Exploiting Bitcoin's Blockchain as a Content Store[C] //Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 1769-1771.

[2] Benet, J., 2014. Ipfs-content addressed, versioned, p2p file system. arXivpreprint arXiv:1407.3561.

[3] Nakamoto, Satoshi. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. Cryptography Mailing list at https://metzdowd.com.

[4] R. A. Canessane, N. Srinivasan, A. Beuria, A. Singh and B. M. Kumar, "Decentralised Applications Using Ethereum Blockchain," 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Chennai, India, 2019, pp. 75-79.

[5] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han and F. Wang, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 11, pp. 2266-2277, Nov. 2019.

[6] Industry Brief: US Consumer Data Breach Report https://www.forgerock.com/resources/view/92170441/industry-brief/us-consumer-data-breach-report.pdf

[7] A. Kiran, S. Dharanikota and A. Basava, "Blockchain based Data Access Control using Smart Contracts," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, 2019, pp. 2335-2339.

[8] Liu, Yuan & Zhao, Zheng & Guo, Guibing & Wang, Xingwei & Tan, Zhenhua & Wang, Shuang. (2017). An Identity Management System Based on Blockchain. 44-4409. 10.1109/PST.2017.00016.

[9] John R. Douceur,“The sybil attack”, Proceedings of the First International Workshop on Peer-to-Peer Systems, pages 251–260, 2002.

[10] Weimin Luo, Jingbo Liu, Jiang Xiong, and Ling Wang, “Defending against whitewashing attacks in peer-to-peer file-sharing networks” In Proceedings of the 4th International Conference on Computer Engineering and Networks, pages 1087–1094, 2015.

[11] S. Rouhani, R. Deters, “Blockchain based access control systems: State of the art and challenges” Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (Web Intelligence ’19). ACM, New York, NY, USA. https://doi.org/10.1145/1122445.1122456

[12] Amit Sahai and Brent Waters, “Fuzzy identity-based encryption”, In Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2005.

[13] Zyskind, Guy & Nathan, Oz & Pentland, Alex. (2015). Enigma: Decentralized Computation Platform with Guaranteed Privacy.

[14] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang and J. Wan, "Smart Contract-Based Access Control for the Internet of Things," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1594-1605, April 2019.

[15] J. P. Cruz, Y. Kaji and N. Yanai, "RBAC-SC: Role-Based Access Control Using Smart Contract," in IEEE Access, vol. 6, pp. 12240-12251, 2018.

[16] Iansiti, Marco; Lakhani, Karim R. (January 2017). "The Truth About Blockchain". Harvard Business Review. Harvard University. Archived from the original on 18 January 2017.

[17] https://en.bitcoin.it/wiki/Genesis_block

[18] https://www.blockchaincongressusa.com/what-is-bitcoin-cash-a-peer-to-peer-electronic-cash-system/

[19] G. Zyskind, O. Nathan and A. '. Pentland, "Decentralizing Privacy: Using Blockchain to Protect Personal Data" 2015 IEEE Security and Privacy Workshops, San Jose, CA, 2015, pp. 180-184.

[20] Zibin Zheng, Hong-Ning Dai, Mingdong Tang, Xiangping Chen, “Blockchain and Trustworthy Systems”, Proceedings of the First International Conference, BlockSys 2019, Guangzhou, China, December 7–8, 2019.

[21] https://www.bitcoinmining.com/how-are-new-bitcoins-created/