Identity ledger on Blockchain

Midterm 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 the 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

With the increasing popularity of computers and mobile devices, it is more and more common for people to use internet-based cloud services to store their personal data like passport, social security number, driver's license. Although these services are easy to use and provide convenience, it poses a high risk in terms of user security and privacy. Malicious users or entities might get their hands on these personal identification documents. In our project, we propose a system, based on blockchain technology, which aims to act as a secure identity ledger for end-users.

II. System Design

A. 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.

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. Moreover, IPFS uses 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 is obtained by running a hashing function on the uploaded documents. 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.

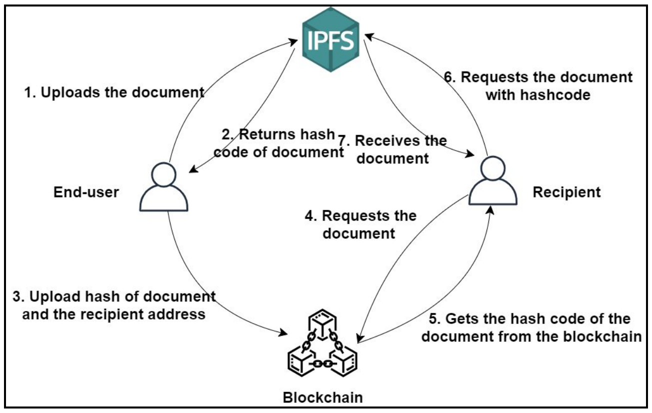
IPFS
1. Uploads the document
2. Returns hash code of document
7. Receives the document
6. Requests the document with hashcode
End-user
Recipient
4. Requests the document
3. Upload hash of document and the recipient address
5. Gets the hash code of the document from the blockchain
Blockchain

Fig. 01: System architecture

B. Blockchain structure

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.

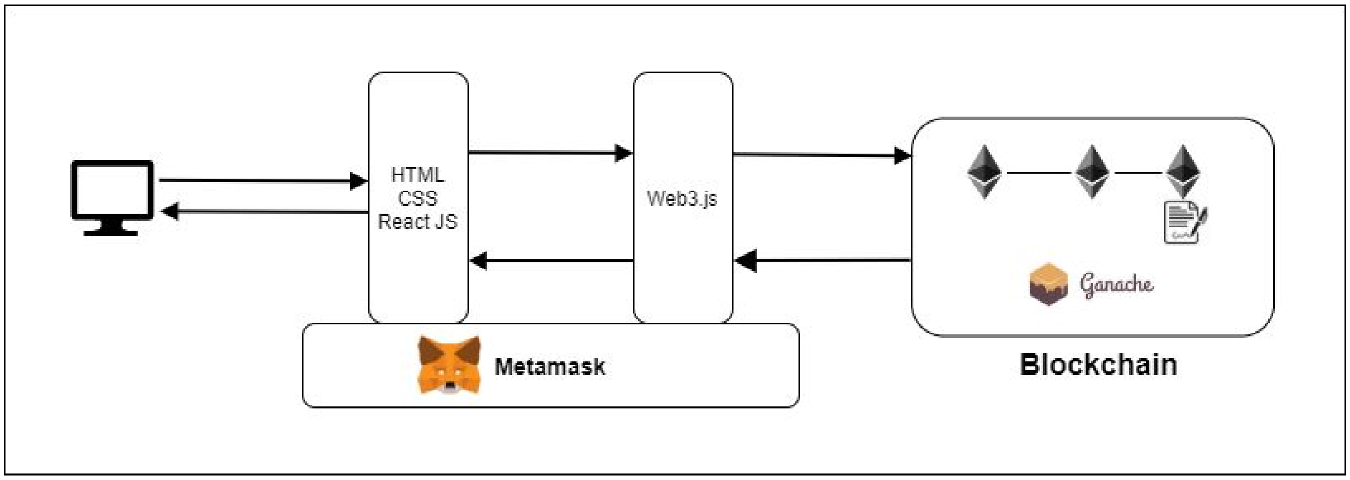

Fig. 02: Application components

III. Implementation

The system implementation by us 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.

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.

IV. Progress and Test runs

We have developed the UI of our application using React Js. We picked React Js for two reasons. First, it has native support for smart contracts written in solidity. Second, having a web application reduces the time and effort needed to deploy traditional desktop-based applications to the end-users. Therefore we decided to go ahead with a web application in order to provide end-users the convenience of using our application using a browser on any device.

Using the IPFS API, we are able to upload a file to the IPFS and retrieve its corresponding hashcode. The user then adds this hashcode and the beneficiary's account address to the blockchain by submitting it to the smart contract. The recipient will be able to retrieve a file from the system if he has been granted access to it. In case the recipient does not have access to a file, nothing is returned from the blockchain. Later on in the project, the requester would be able to download the file by providing the hashcode to IPFS.

We have hosted the developed web application on our localhost. We are using the metamask browser extension to connect the ganache accounts to our web application running on our browser. In all the tests presented in our project, users are the different accounts on the ganache.

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

V. Alternate Approach

At first, we explored the possibility of a desktop-based application in java as we were more familiar with it. We were able to upload and download files to and from the IPFS using the Java API for IPFS. But later on, we found out that React js has native support for the truffle suite, so we decided to go ahead with the development of a web application using React js.

VIII. Conclusion And Future Scope

We have introduced an application for identity management by uploading user's data on IPFS and then storing the IPFS generated hashcode on a blockchain. We are encrypting the hashcode with the recipient's public key before uploading it to the blockchain. On the other end, the recipient is able to retrieve the file's encrypted IPFS hashcode from the application.

In the second half of the semester, we will develop a mechanism by which a user can decrypt the retrieved hashcode with his private key and download the document from IPFS. Moreover, we will explore the possibility to add a digital signature to the uploaded documents to enforce the authenticity of the documents. This will help us address the issue of a recipient duplicating a document after s/he has received it.

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.