Abstract—Identity Management involves the process of identification and authentication of a user to access the services provided by various sectors such as banking, investment, healthcare, government, and online transactions. Identity management not only helps us determine whether a client can access a system but also sets the level of access and authorizations a client has over a system. In the modern digital world, the need for robust digital identity management has never been more important. Most services provided online require certain information from its customers for verification purposes. One of the major concerns of sharing this information with online applications is that the service providers may not store it securely. We have also witnessed few instances in the recent times where the data was compromised without customers knowledge. Hence, there is a need for a system which provides more control to the user which may help prevent security breaches. In this work, we propose the use of blockchain technology based on Ethereum to store hash of identities on a blockchain. The hash of data is generated by the InterPlanetary File System (IPFS) and the corresponding hash is stored on the blockchain. The third party is allowed to request documents from user for identity verification, the user is then notified on the type of document being requested. The user can either share or deny access to the document. The outcome will be greater privacy, control of sensitive data and faster access to services.

Index Terms—Identity Management; Blockchain; Ethereum; Smart Contracts; IPFS

I. INTRODUCTION

With the rapid growth of the internet, most of the services that were provided in a conventional way are now moved online in forms such as online-banking, e-booking, e-mail, and shopping. In order to access the services provided by these online applications, the users must prove their identity by providing a valid identifier. This identifier can take many forms such as personal information contained on a social security card, drivers license, passport, birth certificate, and school or work badges [1]. Identity Management is the process of authenticating, authorizing and identifying an entity (individual or a group of people or the system) to access one or more resources.

Every time a user shares personal information on the online platform to access numerous services, this information is stored in their centralized databases. Users have no control over their information that is stored on various databases. Security is a major concern as hackers are constantly thriving to gain access to such centralized databases.

In order to overcome the security concerns of the centralized database, and to make the system robust, the decentralized approach to identity management is in progress [2]. Blockchain has the potential to remove the third-party interference in the businesses leading to the reduced operational cost. The records stored on the Blockchain are immutable and irreversible providing transparency to the users. As seen in Fig. 1 each block created is linked to the previous block. Also, the data before being stored on the blockchain is hashed. The hash is calculated using the generated hash of the previous block and the data of the current block. The participants use their private key to sign the transactions. With all this it becomes difficult to tamper with the data stored on blockchain. The hacker would have to change the data of the current block as well of the...
previous block to avoid getting detected [9]. Thus with these features it provides more security over the existing solutions for identity management [3].

We propose a modern blockchain based identity management system. This system can be implemented in managing identities of international students, driving licenses or passports, customers in banking or financial sector, etc. A strong identity management solution can solve the problem of centralized server, the hassle of carrying documents for verification. With blockchain, the data would be stored in an encrypted format and in a distributed manner rather than storing it on a single centralized server.

To provide an additional layer of security, we are using Ethereum, a blockchain platform. Ethereum provides us with smart contracts which is compiled when specific conditions are met. The smart contracts hold all the details related to a student. When the student executes the smart contract with credentials along with the digital copies of the hashed documents, the hashed information would be pushed to the blockchain. Once the data is verified by the concerned department the student could share her/his documents and data to any requesting party.

Whenever a third party requests data, the student can see which data has been requested and by whom. With complete control over the data, the student can make sure that data is not being shared to any third party without consent. Also, as the data is stored in a distributed manner, we have better backup in case the server or database goes down. It will help automate all the tasks for the students. Third parties need not ask students for the documents over and again. They can directly request access for the details they need for identity verification through the web application. This solution would save paperwork and time of the university and students.

We organize the rest of the paper by presenting background and background related work in Section II, architecture in Section III, system and implementation details in Section IV and conclusion in Section V.

II. BACKGROUND AND RELATED WORK

Identity management systems authenticate, authorize and identify users. Many private institutes or government organizations need personal information from users to provide them with required services. Traditionally this information is stored in centralized databases. Centralized databases are prone to data breaches. The users have no control over their information. The lack of transparency causes privacy issues in identity verification process [4].

With the emergence of Bitcoin, blockchain technology has started entering all the areas of global business. With its strong features such as distributed ledger, immutability and transparency, blockchain technology is a new approach to identity management systems. Due to decentralization, the users have more control over their identity and there is no centralized authority. In here, we briefly go through some of the technologies mostly used in this domain:

**Ethereum:** is an open-source blockchain platform which supports Smart contracts functionality. It is a network of several nodes which are used to transfer money or store data. Each machine runs an Ethereum client. There is only one main Ethereum network and many test networks. Ether is a cryptocurrency used for paying gas, which is nothing but an operational cost of transaction. Application based on Ethereum is usually referred as Decentralized Application or Dapp [11].

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Ethereum</th>
<th>Hyperledger Fabric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Governing body</td>
<td>Developers, Enterprise Ethereum Alliance</td>
<td>Linux Foundation &amp; IBM</td>
</tr>
<tr>
<td>Cryptocurrency</td>
<td>Ether(ETH)</td>
<td>None</td>
</tr>
<tr>
<td>Ledger type</td>
<td>Permissionless, Public or Private</td>
<td>Permissioned, Private</td>
</tr>
<tr>
<td>Consensus</td>
<td>PoW (Proof of Work)</td>
<td>Pluggable Framework</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Solidity</td>
<td>Chaincode</td>
</tr>
<tr>
<td>Smart contract</td>
<td>Solidity</td>
<td>Chaincode</td>
</tr>
<tr>
<td>Enterprise</td>
<td>Impractical</td>
<td>Practical, Friendly</td>
</tr>
<tr>
<td>Commercial use</td>
<td>B2C purpose</td>
<td>B2B focus</td>
</tr>
</tbody>
</table>

The two blockchain platforms Ethereum and Hyperledger are compared in Table I and concluded that Ethereum has favourable features for the implementation of Identity Management system.

**Smart contracts:** are lines of code, which define set of rules that control the transfer of the assets and manage agreement between users. It has three properties: code, data storage and balance of ether owned by account. The execution of smart contract charges transaction fees in ether, which depend on processing power required. Once the smart contract is deployed it cannot be altered, making transactions secure from unwanted modifications [11].

**Rinkeby:** Rinkeby is a staging network for Ethereum main network. Unlike Ganache, which is a local test network, Rinkeby operates globally. It is used by developers to test and deploy Smart contracts [12].

**Solidity:** It is a JavaScript based language used to write Smart Contracts for Ethereum network. It is intended to center around Ethereum Virtual Machine (EVM). Solidity supports features like inheritance, import of libraries. It uses .sol extension and it has strict typing rules at compile time [12].

**Web3.js:** It is an Ethereum based JavaScript Application Programming Interface(API). It is used to interact with the remote or local Ethereum network using HTTP or IPC connection from JavaScript application [13].

**React:** It is a JavaScript library used to create User Interface. We can create single page web application, or a mobile application to complex web applications. It is easy to customize user interface of a website using react as you can create your tags for particular functionality [14].
**Metamask:** It is a chrome extension used to interact with the Ethereum network. It turns a normal browser into a web3 browser letting websites retrieve data from blockchain and letting users securely manage identities and sign in. It can also be used to store the data. One user can create many accounts and all the accounts are encrypted and securely stored within your browser [15].

**MongoDB:** MongoDB is a non-relational database management system. It uses document as its basic data structure which is comprised of field and value pairs. MongoDB documents use Binary JSON. [16].

**IPFS:** InterPlanetary File System (IPFS) is a decentralized database built on top of MongoDB. With a peer-to-peer implementation, it avoids single point of failure and protects data from denial of service attacks [17].

Among recent solutions, uPort is a decentralized identity framework. The users can create their own identity using uPorts mobile application. This application is built using Ethereum and IPFS. Each uPort identity corresponds to two smart contracts- controller and proxy. For creating a new identity uPort generates an asymmetric key pair. The controller contract is created, and it saves a reference to the public key and initiates a transaction to Ethereum. For storing a reference to controller new proxy is created. The address of the proxy consists of a unique uPort identifier called uPortId. The Private key associated with uPortId is stored on users device. The user can nominate trustees to retrieve the key if it is lost. Once information provided by the user is verified, uPort stores JSON object of that data onto IPFS. A reference to this off-chain data is stored on blockchain using registry contract which can be accessed by each uPort identity. Any third-party client application can request data to the user. If the user accepts the request, a signed transaction is sent to Ethereum and information can be accessed by the client application [5].

**III. Architecture**

A decentralized and distributed or a blockchain network consists of nodes that are organized together to store, share, update and keep track of blockchains (sometimes also called ledger or digital transaction data). The nodes of the blockchain network in Fig. 2 may include computing devices or systems such as PCs, laptops, servers, computer farms or clusters, virtual or cloud systems, smartphones or tablets. These nodes may carry the entire blockchain, or some part/reduced version of it and may be able to trace back to verify historical transactions and blocks on the blockchain.

Any blockchain node may keep an exact copy of the blockchain and hence may verify and audit any existing or newly added transactions on the blockchain by validating the hash values on the blocks and digital signatures on the transactions. However, adding a new block to the blockchain may need the consensus of the blockchain network. In particular, a consensus scheme that is secure against double spending or double counting dispute may be employed by the blockchain network. For instance, the blockchain network should not let a student to get two identities (possibly cloned) or fake records.

The next block containing transaction records that will be linked to a blockchain is decided by the blockchain network using some consensus scheme including the consensus of the network majority (e.g. proof-of-work); a consensus protocol where the creator is chosen in a deterministic way (e.g. Casper or other proof-of-stake schemes) or some other Byzantine Fault Tolerance consensus schemes. All these schemes are designed to be secure against double spending or double counting problems. However, sometimes the data written to the blockchain could be erroneous due to some typos, inaccurate readings or human errors. Some authorized entity may correct such errors by creating some revision transactions that may be placed into the current or coming blocks referring the erroneous transaction and a reasoning for the revision.

In the identity management system, the users can add their identity on a blockchain through our web-based application which acts as an interface among user, third party and the database as seen in Fig. 2. When a user registers on the Ethereum network, the user will receive an Ethereum account address. This address can be used for verification purposes in the future. The users can then use the account address to upload their details and documents through the web application. The transaction would be posted to the blockchain and data would be updated in the file system.

When the third-party requests access to the user data, the request is sent to the concerned user. The user can view the request and see what document has been requested by the requesting party. On users approval, the generated hash would be sent to the requesting party. The requesting party can use the hash to view all the data related to the user. The users can also update their own data by logging into the system using necessary credentials.

There is always a danger of unauthorized access and manipulation of database in the traditional identity management system, the database architecture of Blockchain overcomes the shortcomings of the traditional database. The modules
involved in the back-end of our architecture are Blockchain Database API, Database and File System (Fig. 3). The functionality of these modules is as follows:

- **Blockchain Database API**: Using the nonce and hash of previous transaction Blockchain Database API will calculate the hash value for the transaction. Nonce is just a random string. Blockchain Database API will make an entry of the transaction, nonce, and hash into the database. To detect and prevent any unauthorized change, Blockchain Database API will again calculate the hash value based on the information present in the database. The API can be notified if any change is made as the hash value will change. In this way, we can ensure the integrity of the data.

- **File System**: Like every other technology, Blockchain has its own drawbacks. It is very expensive to store all the data on Ethereum blockchain. Each operation in the Ethereum Virtual Machine consumes gas. For example: ADD/SUB operation consumes 3 units of gas and MULTIPLY/DIVIDE operation consumes 5 units of gas. The proof of work in the recent days have slowed down the speed of transactions. Cryptokitties brought Ethereum to halt due to heavy load on the network resulting in expensive transactions. To minimize the cost on storage of the data on Blockchain, we still need external data storage. Hence, Blockchain is coupled with IPFS for storage of data. The IPFS [6] is a peer to peer protocol which controls how data is stored and shared. It allows the user to store the files on the system and a cryptographic hash is created for each file on the system. Anyone with the hash can view the content of the file. In this study, we are utilizing the distributed IPFS to store the student details and their documents such as drivers license, I-20 document, Photo ID, Social security card. The hash of student information and files along with student ID is stored on the blockchain. Hence, whenever any department requests an access to view a valid identity of student data, the hash of the file will be provided to the requesting party.

- **Database**: It is an organized collection of data which is stored and accessed electronically. It provides high throughput with low latency. It also has powerful query functionality, decentralized control and immutable data storage.

### IV. SYSTEM DESIGN

In identity management system, the user can sign-up to the web application by entering the information and the user is also expected to choose a role from among ones given on the website such as Student, Validator and Requestor as seen in Fig. 4. The user can then sign in to the application to access the services. The username and password entered by the user is checked against the login credentials stored in the database. Once the authentication is successful, the user is redirected to the respective home page.

1) **Student**: In case of a student, the page has three functionalities.

   - **Prompts the student to upload the identity documents**: When a student uploads the document on the website, it will be stored on IPFS [7] and a corresponding hash of the document is generated. Also, the smart contract is triggered automatically, the hash of this document is stored on the student’s blockchain address.

   - **Option to fetch the hash of existing documents stored on blockchain**: The students are provided with the Fetch Hash button to retrieve hash of the documents, which will trigger an event and return hash of all the documents along with the name of the document as key.

   - **Table of identity verification documents requested by third parties**: The students also get the request from third parties for their identity verification. If the student accepts the request, he can fetch the hash from the second functionality as explained above and send it to the requestor. Otherwise, the request can be rejected.

2) **Validator**: In case of a validator, they can view all the documents uploaded by each student with the status being pending. The validators are the authorities in college and each document will be sent to the respective authority for validation. The authorities can validate the authenticity of the document and either Approve or Disapprove the document by clicking on the respective button. Once the process is completed, the status of the document will be reflected on Students Home page.
3) **Requestor:** In case of a Requestor, any third party such as Bank are provided with an option to request documents from the student through the web application using a unique key - Student’s unique Metamask Address. The requestor can select from the options, the type of document that is needed by them. The list of the documents requested is displayed to the Students on their Home page. The requestor page also has an option to view the documents shared by students. Each document is marked as either Approved or Not approved. If the document is approved, the requestor can proceed with the next steps otherwise they can ask student to submit the verified copy of the documents and hold the procedure till they receive a valid copy.

**A. User Interface Implementation**

The user interface is developed using programming languages such as ReactJS and NodeJS. ReactJS along with Bootstrap 4 and jQuery is used for web UI development. The advantage of using react over other web-based technologies is that it provides interactive user interface with customizable components. NodeJS is used for developing Application Programming Interface (API) to interact with the database (Backend). In this study, Mongo DB which is a NoSQL database is used, as it allows unstructured data to be stored in the database. This feature is particularly advantageous while storing information of a student which can vary from name, address, email id (string type) to documents (.pdf or .jpeg format). The script is written in such a way that the data is inserted to the database in json format. Every entry will have id associated with it. The data can be fetched from the collection using this unique id. Separate tables are maintained for users with different roles (validator, student and requesting third party). To install all the packages node packet manager (npm) is used. The important packages which are needed are: bcrypt for encrypting the delicate information like password, studentid etc, body-parser for fetching the data in json format, cors is used to avoid cross origin resource sharing error, express is a NodeJS framework which provides robust features for web or mobile application, mongoose is used to interact with mongodb, jsonwebtoken is a method used to securely decode, verify and generate JWT, "nodemon" is used for watching the api written to interact with database, if any changes are made to the file, it is automatically restarted.

**B. Smart Contract Implementation**

We have developed a smart contract-User.sol in Solidity language. This contract [8] manages the data stored on the blockchain. Whenever the student uploads any document through our application, the document is stored in IPFS and IPFS-hash is generated. The sendHash function in this contract is invoked. It takes IPFS-hash and the name of the document as arguments from the back-end and pushed onto the blockchain. The hasUser function checks whether the record of account address which is invoking the contract has already been pushed on the blockchain. The getIPfsHashByAddress function is written to retrieve the stored details of the student on the blockchain. It takes the users account address as a parameter and invokes hasUser function. If the user is present on the blockchain, this function returns IPFS-hashes and names of the documents which belong to that user.

**V. Conclusion**

To conclude, this paper explains how blockchain technology can be applied to identity management for improving security. The web application takes information from the student and stores it in the IPFS database. The hash generated from the information is securely stored on the Ethereum blockchain along with students Ethereum address. Any third-party can request data from the student. The student has the right to approve or reject the request. If the request is approved information is passed to the third-party via blockchain. To alleviate the growing concerns for information privacy, this system allows students to have complete control over their data.

**REFERENCES**


