ABSTRACT

The blockchain is a distributed network that records digital transactions on a publicly accessible ledger. This paper explores whether blockchain technology is a suitable platform for the preservation of digital documents. This paper suggests that the blockchain's hash functions offer a better strategy for accountable preservation of documents. Compared to digital certificates, hashing provides better privacy and security. It is a form of authentication that does not require trust in a third-party authority, and the distributed nature of the blockchain network removes the problem of a single point of failure.

This application will provide following functionalities

- Hashing of content of document
- Hashing of authors email.
- Time stampping of document.
- Storage on Ethereum blockchain.

LIST OF FIGURES

LIST OF TABLES

Table of Contents

INTRODUCTION

The blockchain has been with us since 2009. In its seven years of existence, it has successfully resisted attempts to hack into it, take it down or co-opt it. While the technology is at a crossroads in its development, there are many use cases for its adoption across industry, including the field of records management. The notarization of electronic records presents novel challenges for records managers. In a paper records environment, the creator of a record states ownership of the document, or assents to an agreement articulated in the document, by signing or countersigning it. From the records manager's perspective, the document is the property of the party that signed it. The signature is synonymous with the document.

Thus using the concept of decentralization, proof of work and time stamping features of blockchain the authorization and accountability which in addition with hashing provides a way to preserve the content of documents, make author accountable and store the time of document preservation.

The project based on the above concept reads the content of the document which is uploaded instead of scanning the document as an image. Content once read is then hashed and that information is uploaded on blockchain network instead of the original content which makes the content safe. This process thus ensures safety of content even if it is being uploaded on a public distributed ledger.

OVERVIEW

PROPOSED WORK

In this project, we present a way to store and authenticate the content and authorship of documents respectively. The process involves uploading the document which needs to be secured. Once uploaded the document's content is hashed using the concept of hashing. Similarly the email of author is also hashed. Both these hashes are joined together and hashed again. This hashing is then stored on blockchain along with the time stamp. Once the blockchain approves the upload then it becomes the part of blockchain forever.

All the steps described above are implemented using the Ethereum framework. Since ethereum framework is used for the project the concept of Smart Contract is also implemented. In this project thus all the middleman work is done by smart contracts. All the transactions are implemented by smart contract. This reduces the human interaction between any transactions and thus also reduces significantly the chance of any malicious interference in the process.

Therefore if implemented in a correct way the proposed project will be able to provide a way to secure a document and make its author accountable and help in avoidance and detection of corporate fraud and espionage.

Flow Chart

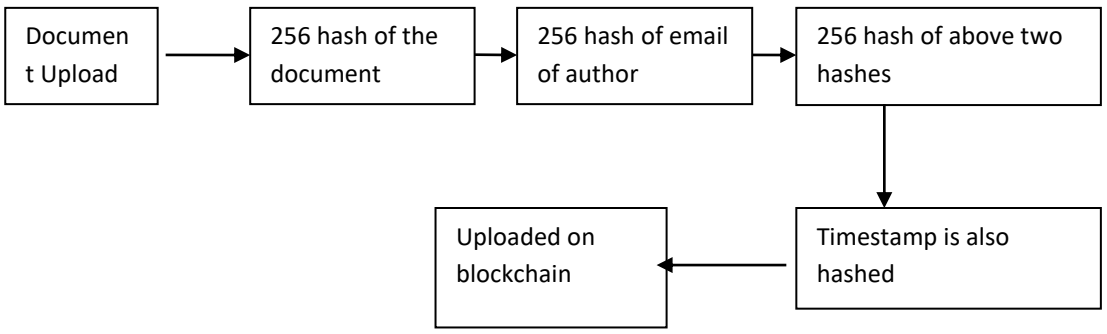


Figure 1: Flow Chart diagram of working of project

SOFTWARE AND HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS

- METAMASK
- GOOGLE CHROME
- Operating system (windows, Linux etc.)
- TRUFFLE
- BLOCKCHAIN
- Node.js

HARDWARE REQUIREMENTS

- Ram (minimum 3 GB, 8 GB recommended; plus 1 GB for the Android Emulator (if used.))
- Disk Space:
    - ✓ 500 MB disk space for Android Studio.
    - ✓ At least 1.5 GB for Android SDK, emulator system images, and caches.
- 1280 x 800 minimum screen resolution.
- Android phone with version 4.1(jelly bean) or higher.

TECHNOLOGIES USED

ETHEREUM FRAMEWORK

Ethereum is an open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality.[3] It supports a modified version of Nakamoto consensus via transaction based state transitions.

Ether is a cryptocurrency whose blockchain is generated by the Ethereum platform. *Ether* can be transferred between accounts and used to compensate participant mining nodes for computations performed.

The DAO event

In 2016 a decentralized autonomous organization called The DAO, a set of smart contracts developed on the platform, raised a record US$150 million in a crowdsale to fund the project.[40] The DAO was exploited in June when US$50 million in ether were claimed by an anonymous entity.[41][42] The event sparked a debate in the crypto-community about whether Ethereum should perform a contentious "hard fork" to reappropriate the affected funds.[43] As a result of the dispute, the network split in two. Ethereum (the subject of this article) continued on the forked blockchain, while Ethereum Classic continued on the original blockchain.[44] The hard fork created a rivalry between the two networks.[45]

Virtual Machine

The Ethereum Virtual Machine (EVM)[63][64] is the runtime environment for smart contracts in Ethereum. It is a 256-bit register stack, designed to run the same code exactly as intended. It is the fundamental consensus mechanism for Ethereum. The formal definition of the EVM is specified in the Ethereum Yellow Paper.[54][65] It is sandboxed and also completely isolated from the network, filesystem or other processes of the host computer system. Every Ethereum node in the network runs an EVM implementation and executes the same instructions. In February 1, 2018, there were 27,500 nodes in the main Ethereum network.[66] Ethereum Virtual Machines have been implemented in C++, Go, Haskell, Java, JavaScript, Python, Ruby, Rust, and WebAssembly (currently under development).[67][68] The Ethereum-flavoured WebAssembly (dubbed "e-WASM") is expected to become a major component of the "Web 3.0", a World Wide Web where users interact with smart contracts through a browser

Applications

Ethereum blockchain applications are usually referred to as DApps (decentralized application), since they are based on the decentralized Ethereum Virtual Machine, and its smart contracts.[5] Many uses have been proposed for Ethereum platform, including ones that are impossible or unfeasible.[74][75][48] Use case proposals have included finance, the internet-of-

things, farm-to-table produce, electricity sourcing and pricing, and sports betting.[48][76] Ethereum is (as of 2017) the leading blockchain platform for initial coin offeringprojects, with over 50% market share.[77]

As of January 2018, there are more than 250 live DApps, with hundreds more under development

In this project Ethereum Framework is used to develop our application and this platform is used to deploy the application on ethereum blockchain.

BLOCKCHAIN

A **blockchain**,[1][2][3] originally **block chain**,[4][5] is a continuously growing list of records, called *blocks*, which are linked and secured using cryptography.[1][6] Each block typically contains a cryptographic hash of the previous block,[6] a timestamp and transaction data.[7] By design, a blockchain is inherently resistant to modification of the data. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way".[8] For use as a distributed ledger, a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks. Once recorded, the data in any given block cannot be altered retroactively without the alteration of all subsequent blocks, which requires collusion of the network majority.

Structure[edit]

A blockchain is a decentralized, distributed and public digital ledger that is used to record transactions across many computers so that the record cannot be altered retroactively without the alteration of all subsequent blocks and the collusion of the network.[1][29] This allows the participants to verify and audit transactions inexpensively.[30] A blockchain database is managed autonomously using a peer-to-peer network and a distributed timestamping server. They are authenticated by mass collaboration powered by collective self-interests.[31] The result is a robust workflow where participants' uncertainty regarding data security is marginal. The use of a blockchain removes the characteristic of infinite reproducibility from a digital asset. It confirms

that each unit of value was transferred only once, solving the long-standing problem of double spending. Blockchains have been described as a value-exchange protocol.[21] This blockchain-based exchange of value can be completed more quickly, more safely and more cheaply than with traditional systems.[32] A blockchain can assign title rights because it provides a record that compels offer and acceptance.[1]

Decentralization**[edit]**

By storing data across its peer-to-peer network, the blockchain eliminates a number of risks that come with data being held centrally.[1] The decentralized blockchain may use ad-hocmessage passing and distributed networking.

Peer-to-peer blockchain networks lack centralized points of vulnerability that computer crackers can exploit; likewise, it has no central point of failure. Blockchain security methods include the use of public-key cryptography.[4]:5 A *public key* (a long, random-looking string of numbers) is an address on the blockchain. Value tokens sent across the network are recorded as belonging to that address. A *private key* is like a password that gives its owner access to their digital assets or the means to otherwise interact with the various capabilities that blockchains now support. Data stored on the blockchain is generally considered incorruptible.[1]

While centralized data is more easily controlled, information and data manipulation are possible. By decentralizing data on an accessible ledger, public blockchains make block-level data transparent to everyone involved.[45]

Every node in a decentralized system has a copy of the blockchain. Data quality is maintained by massive database replication[9] and computational trust. No centralized "official" copy exists and no user is "trusted" more than any other.[4] Transactions are broadcast to the network using software. Messages are delivered on a best-effort basis. Mining nodes validate transactions,[33] add them to the block they are building, and then broadcast the completed block to other nodes.[35]:ch. 08 Blockchains use various time-stamping schemes, such as proof-of-work, to serialize changes.[46] Alternate consensus methods include proof-of-stake.[33] Growth of a

decentralized blockchain is accompanied by the risk of node centralization because the computer resources required to process larger amounts of data become more expensive.[47]

Types of blockchains[edit]

Currently, there are three types of blockchain networks - public blockchains, private blockchains and consortium blockchains.[145][*self-published source?*]

Public blockchains**[edit]**

A public blockchain has absolutely no access restrictions. Anyone with an internet connection can send transactions[*disambiguation needed*] to it as well as become a validator (i.e., participate in the execution of a consensus protocol). [146][*self-published source?*] Usually, such networks offer economic incentives for those who secure them and utilize some type of a Proof of Stake or Proof of Work algorithm.

Some of the largest, most known public blockchains are Bitcoin and Ethereum.

Private blockchains**[edit]**

A private blockchain is permissioned.[58] One cannot join it unless invited by the network administrators. Participant and validator access is restricted.[7][*unreliable source?*]

This type of blockchains can be considered a middle-ground for companies that are interested in the blockchain technology in general but are not comfortable with a level of control offered by public networks. Typically, they seek to incorporate blockchain into their accounting and record-keeping procedures without sacrificing autonomy and running the risk of exposing sensitive data to the public internet.

Consortium blockchains**[edit]**

A consortium blockchain is often said to be semi-decentralized. It, too, is permissioned but instead of a single organization controlling it, a number of companies might each operate a node on such a network. The administrators of a consortium chain restrict users' reading rights as they see fit and only allow a limited set of trusted nodes to execute a consensus protocol.[147]

Solidity

Solidity is a contract-oriented programming language for writing smart contracts.[1] It is used for implementing smart contracts[2] on various blockchain platforms.[3][4][5] It was developed by Gavin Wood, Christian Reitwiessner, Alex Beregszaszi, Liana Husikyan, Yoichi Hirai and several former Ethereum core contributors to enable writing smart contracts on blockchain platforms such as Ethereum.[6][7][8] Solidity is a statically-typed programming language designed for developing smart contracts that run on the EVM.[14][15] Solidity is compiled to bytecode that is executable on the EVM. With Solidity, developers are able to write applications that implement self-enforcing business logic embodied in smart contracts, leaving a non-repudiable and authoritative record of transactions.[16][17] Writing smart contracts in smart contract specific languages such as Solidity is referred to as easy (ostensibly for those who already have programming skills).[18]

As specified by Wood it is designed around the ECMAScript syntax to make it familiar for existing web developers; unlike ECMAScript it has static typing and variadic return types. Compared to other EVM-targeting languages of the time such as Serpent and Mutan, Solidity contained a number of important differences. Complex member variables for contracts including arbitrarily hierarchical mappings and structs were supported. Contracts support inheritance, including multiple inheritance with C3 linearization. An application binary interface (ABI) facilitating multiple type-safe functions within a single contract was also introduced (and later supported by Serpent). A documentation system for specifying a user-centric description of the ramifications of a method-call was also included in the proposal, known as "Natural Language Specification".[19][20]

SOFTWARE REQUIREMENT ANALYSIS

The advent of blockchain technology provided us an opportunity to divulgue more into benefits of decentralized network and distributed ledgers. Ethereum platform provided us a way to develop decentralized applications which it uses with the concept of Smart Contracts.

Functionality:

- As soon as anyone opens the application it shows fields for entry of email of author.
- Also it have space for uploading of document.
- Once that is done then the hash are calculated.
- Hashes calculated are:
    1. Hash of content of document.
    2. Hash of authors email with hash calculated above.
    3. Hashing of timestamp and hash calculated in previous step.
- The documents are then uploaded on blockchain.

All the above functionalities are provided by the help of contracts which are written in solidity.

Below the various concepts which are used are discussed in detail.

A.HASHING

Hashing is a form of document authentication in which documents are not signed directly. Instead, a hash function generates a hash value to confirm that the authentication of the digital signatures has taken place. There are two components to hashing:

- The hash function is a hexadecimal algorithm, such as SHA-256, that maps an input data of any size into a uniform, usually compressed, file size. In digital preservation, hash functions confirm that no changes have been made to a digital document.
- The hash value is the output of a specific length that permanently identifies the input data (Pedro, 2015, p.95).

The hash work is a restricted procedure. This implies the client can make the hash from input information, yet not utilize the hash to uncover the information. Should a records chief modify even one piece from the info information and afterward attempt to apply a similar hash work, the administrator will create totally different hash esteem.

B. TIMESTAMPING

A timestamp demonstrates that a specific dataset existed at one point in time (Pedro, 2015, p. 99). The blockchain strategy makes time stamped hinders through distributed innovation, subsequently disintermediating Time Stamping Authorities (TSAs). Diggers on the Bitcoin blockchain timestamp each square which contains ten minutes of exchanges. The diggers are, viably, working as a dispersed TSA. This implies there is no requirement for intermittent re-timestamping of marks because of lapsing keys. In limited time materials for its new BLT cryptographic calculation, the product security organization Guard time expressed the time and trustworthiness of the mark can be demonstrated numerically, without dependence on the security of keys or of confided in parties (Guard time, 2016). Amanti (2016) expressed that the time it takes for a TSA to check a move is estimated in seconds, though the blockchain's confirmation takes minutes (para. 23). He additionally noted two different focal points of blockchain timestamping over TSA timestamping:

- Long-term preservation can be achieved without the maintenance costs that come with a TSA-issued certificate (Amati, 2016, para. 24). 10 ‖ See Also: Vol. 3 (Spring 2017)
- Archivists can exploit the convenience of verifying the signature with the document and public key without having to safeguard the digital signature on a central server (Amati,2016)

C.SMART CONTRACT

A smart contract is a PC convention planned to carefully encourage, check, or uphold the arrangement or execution of an agreement. Smart contracts permit the execution of tenable exchanges without outsiders. These exchanges are traceable and irreversible. Smart contracts were first proposed by Nick Szabo, who instituted the term, in 1994.

Defenders of smart contracts assert that numerous sorts of legally binding provisions might be made incompletely or completely self-executing, self-upholding, or both. The point of smart contracts is to give security that is better than customary contract law and to diminish other exchange costs related with contracting. Different cryptographic forms of money have actualized sorts of smart contracts.

Ethereum enables engineers to program their own particular shrewd contracts, or 'self-sufficient specialists', as the ethereum white paper calls them. The dialect is 'Turing-finished', which means it bolsters a more extensive arrangement of computational directions and along these lines in this task custom savvy contracts are made.

Smart contracts can:

- Function as 'multi-signature' accounts, so finances are spent just when a required level of individuals concur.
- Manage understandings between clients, say, on the off chance that one purchases protection from the other.
- Provide utility to different contracts (like how a product library works).
- Store data around an application, for example, area enrolment data or participation records.

SOFTWARE DESIGN

DFD DIAGRAMS

The Data flow diagram can be explained as the separate levels indicating the individual complexity in the each level of the system and gives a detailed explanation in the further levels that are following them.

1) Level 0

Initially in the first level of the Data flow the level 0 explains the basic outline of the system.

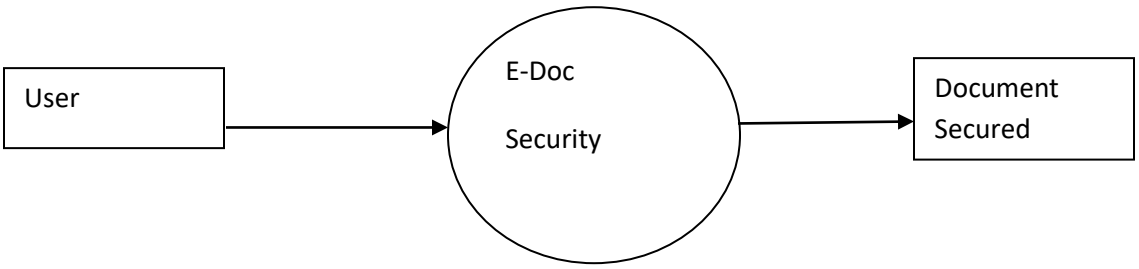
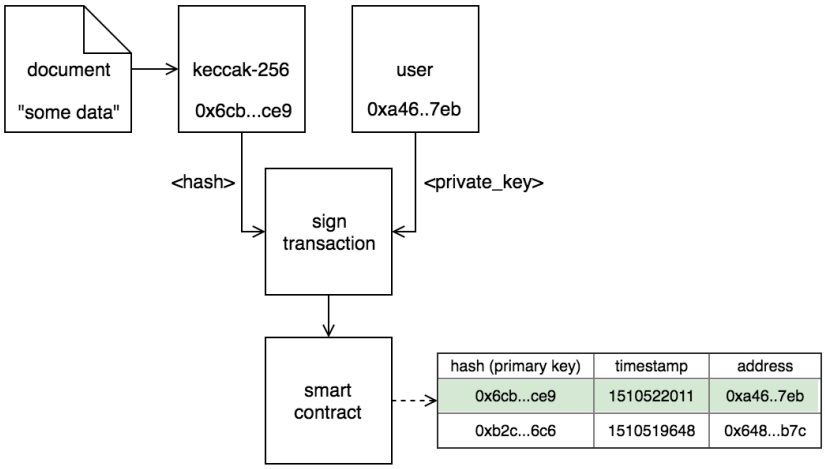


Figure 1: Level 0 DFD

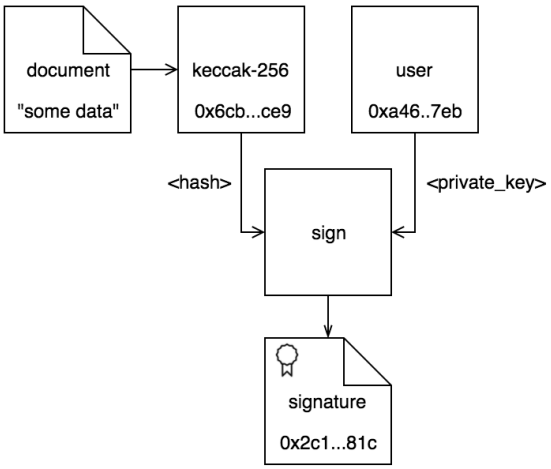
2) Level 1

The level 1 of the Data flow diagram given explains in detail about the process of how the user is able to secure the content and ownership of documents .

Stamping a document



Generating signature



Verifying signature

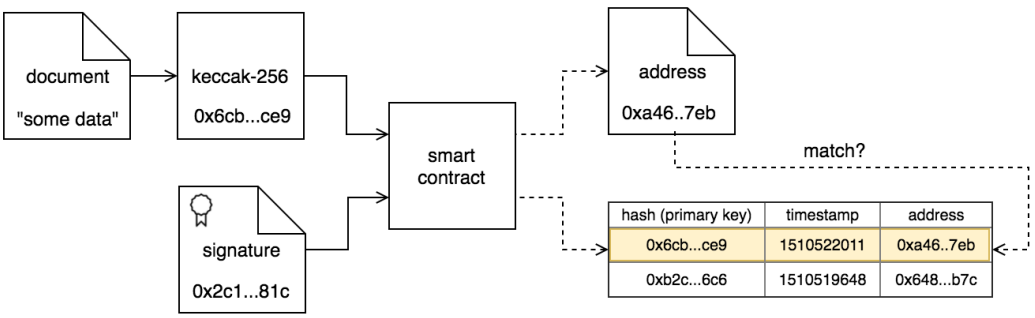


Figure 2: Level 1 DFD

USECASE DIAGRAM

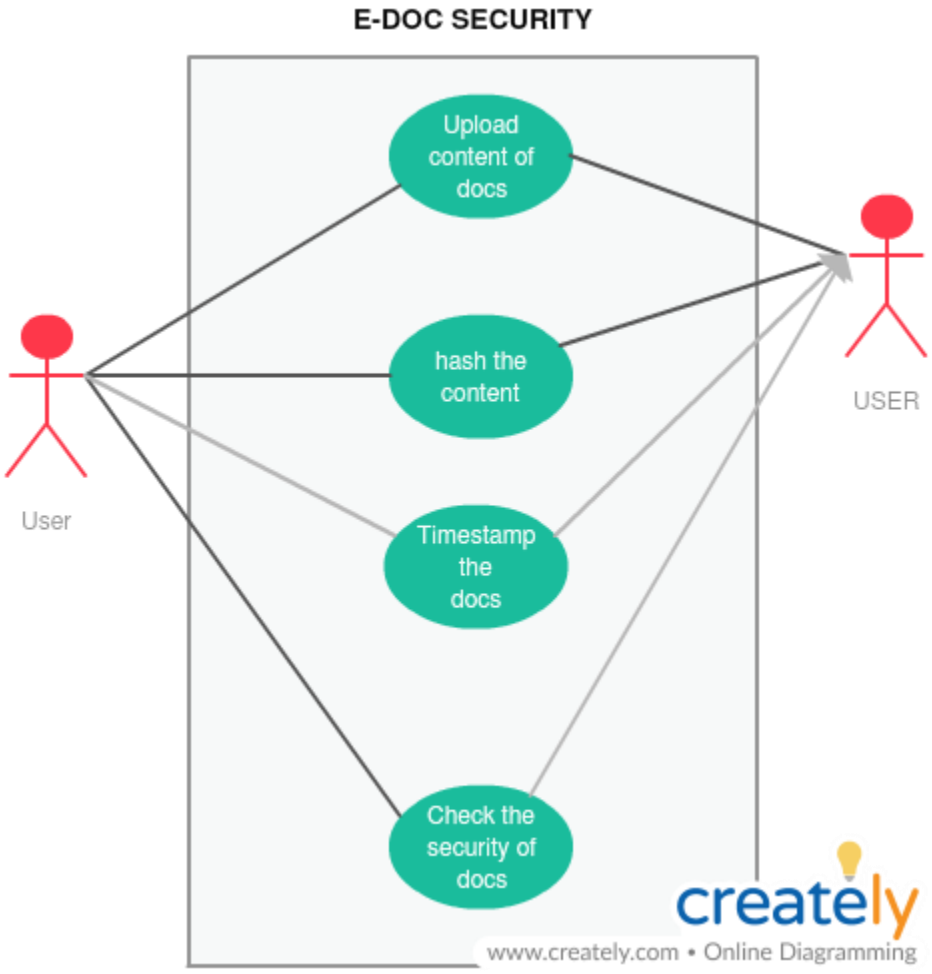


Figure 3: USECASE DIAGRAM

OUTPUTS:

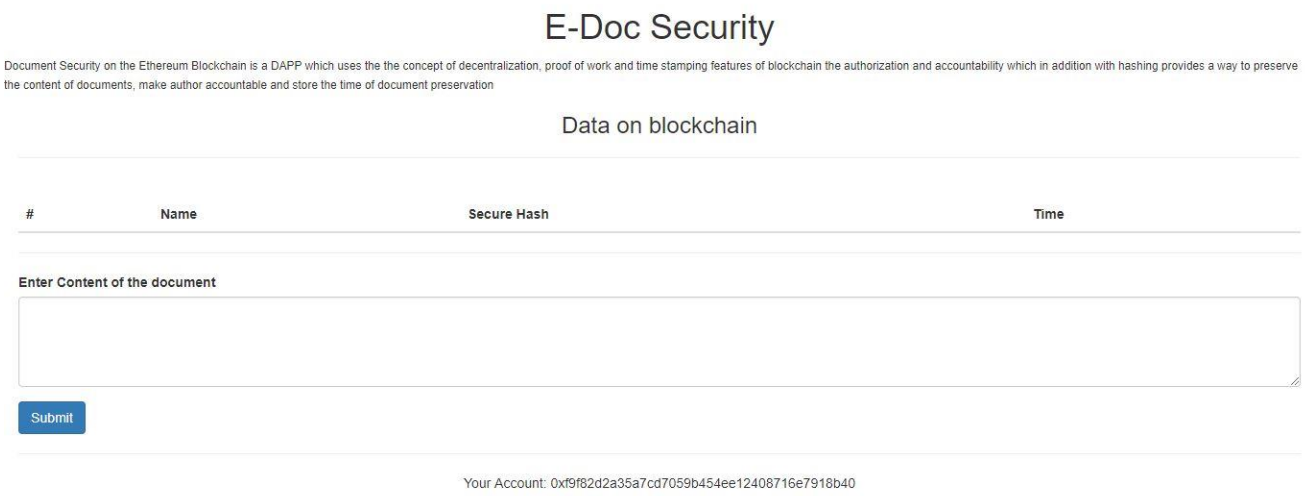


Figure 4: On this page any individual can with account on blockchain use E-DOC SECURITY .

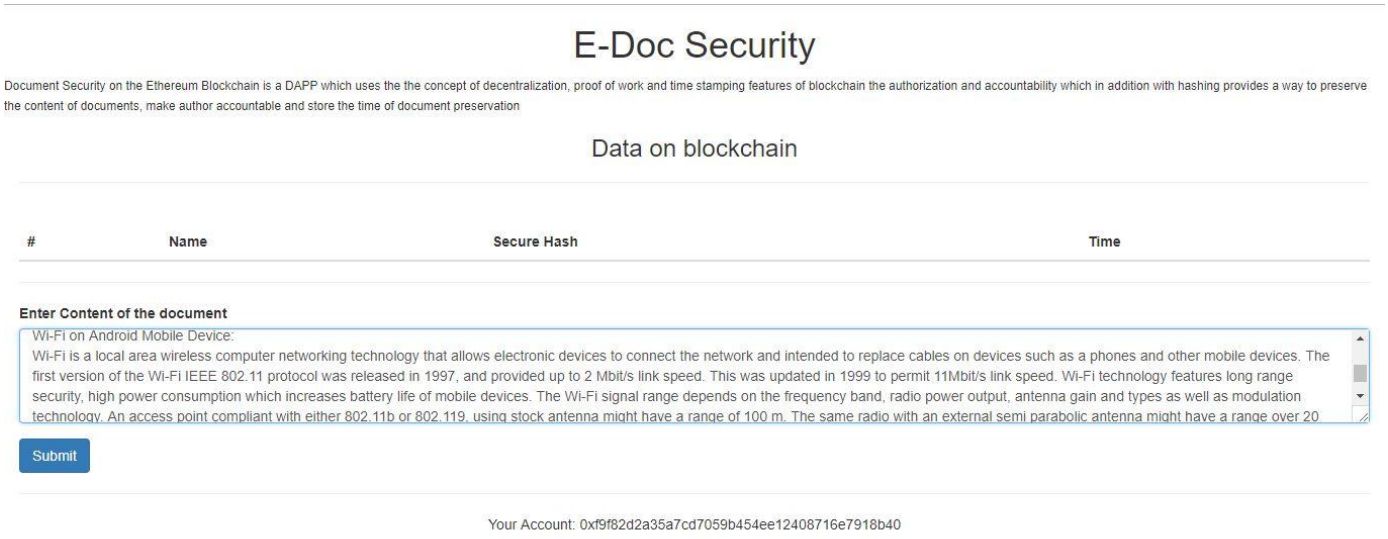


Figure 5: On this page content of document is copied.

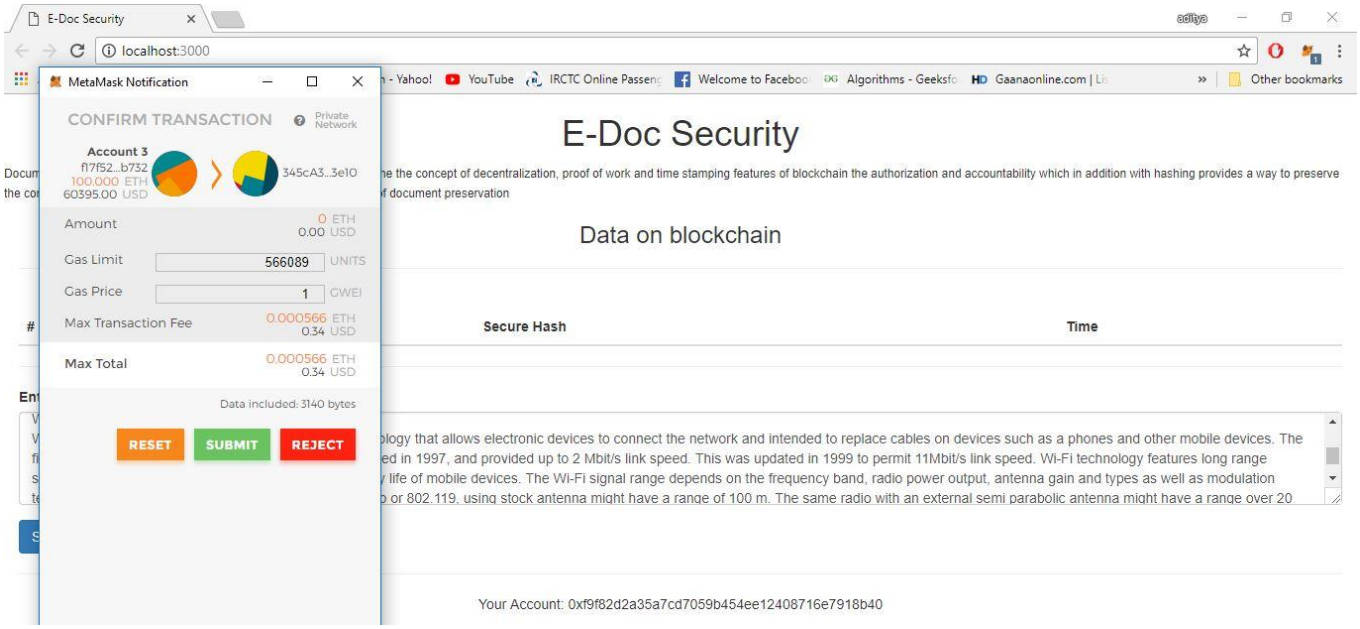


Figure 6: On this page blockchain transaction is confirmed.

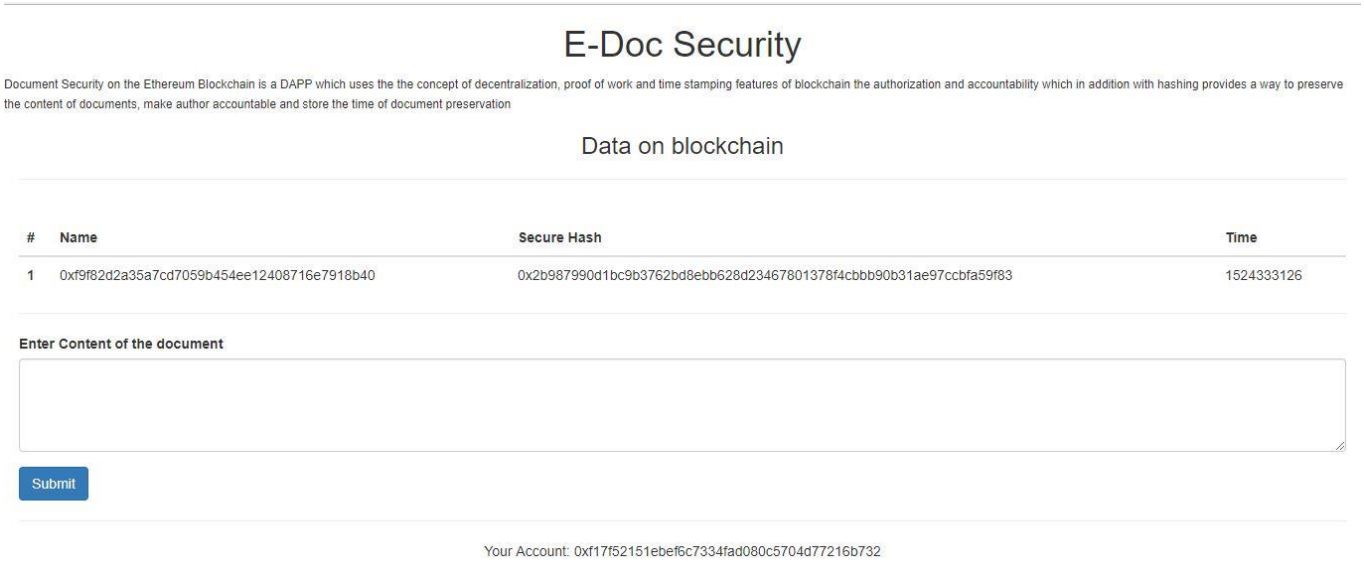


Figure 7: On this page hash along with ownership and timestamp is stored on blockchain.

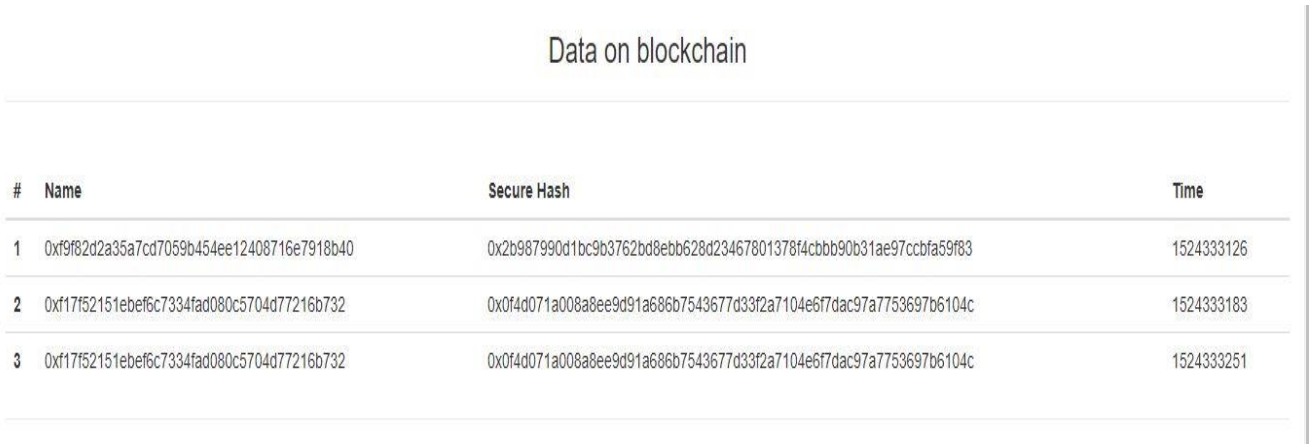


Figure 8: On this page same content with same ownership is hashed but with different timestamp.

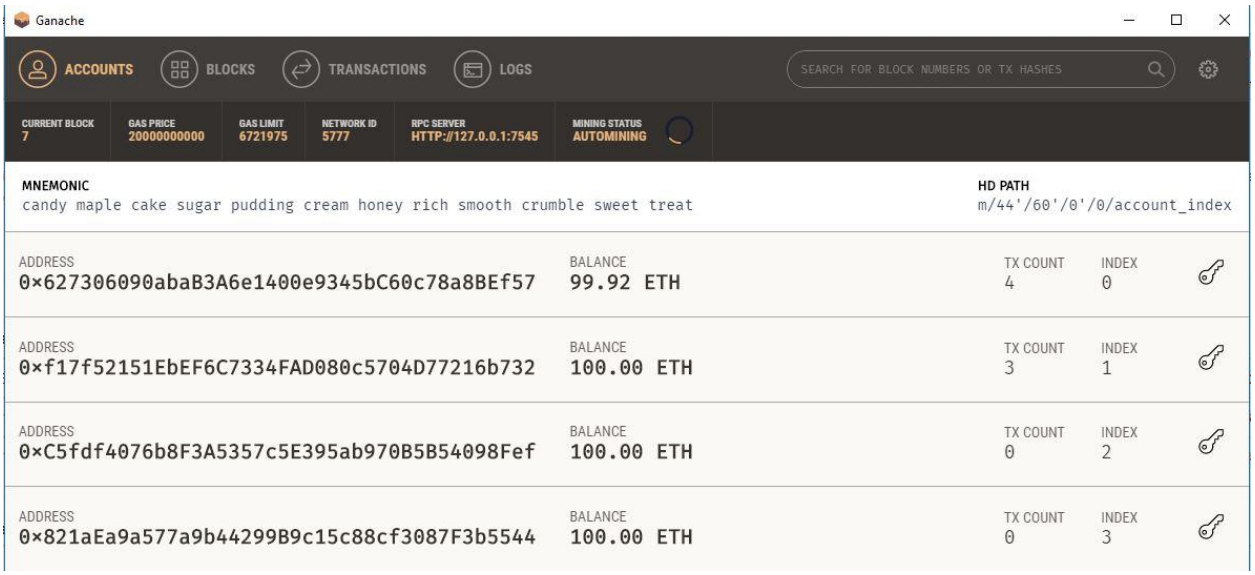


Figure 9: On this page details of blockchain accounts.

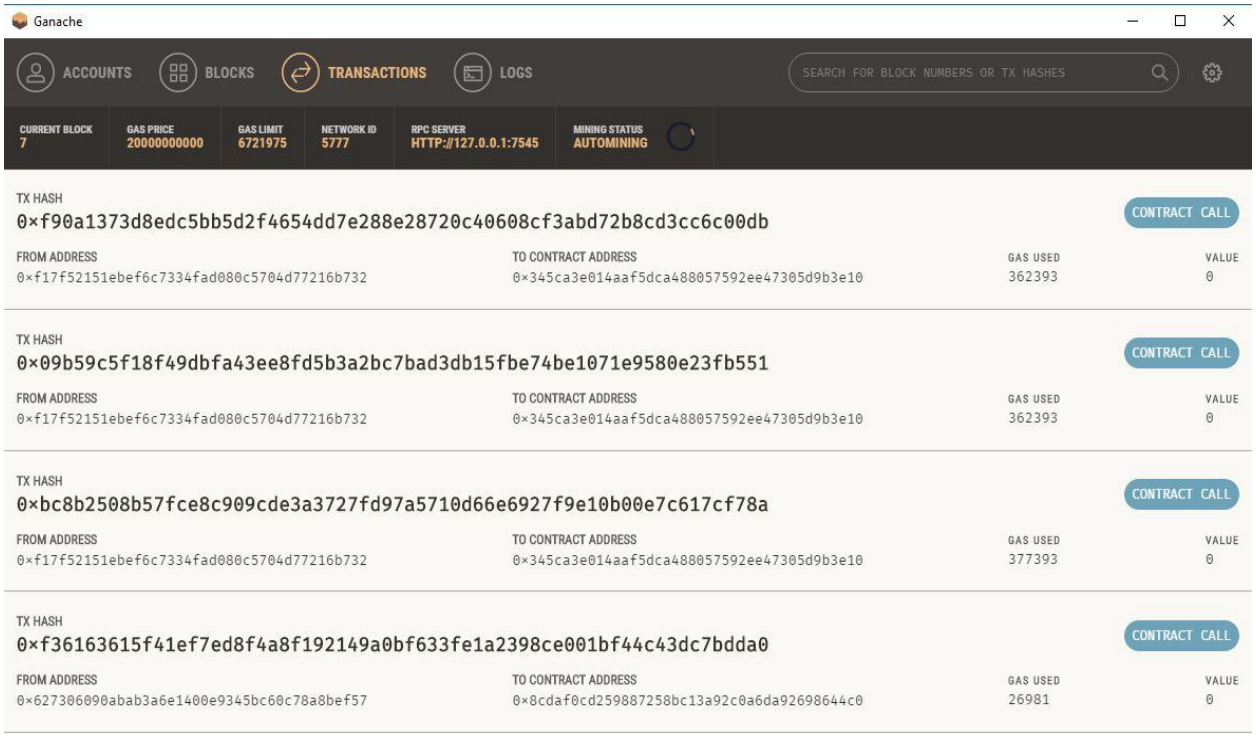


Figure 10: On this page details of transactions on blockchain.

CONCLUSION

SUMMARY OF WORK DONE

The proposed system E-Doc Security is the way of authenticating and stamping the e-docs. The disadvantages of using the conventional way included involvement of middle man or human notary which is susciptle to corruption and thus cannot be trusted complelely. All these disadvantages has been removed from the proposed system.

This project proposes the idea of using blockchain technology and smart contracts to securely authenticate and store the content and authorship of documents along with time they were published so as to avoid and detect fraud and reduce the chances of government or corporate espionage.

The proposed system is developed in such a way that they can be deployed to various devices of which have browsers and installed the ethereum blockchain viewer software..

SCOPE OF FUTURE ENHANCEMENT

The E-Doc Security can be further developed into system with the following enhancements:

i. The future work can be extended to using image processing to scan the documents instead of uploading them for their content and using features of image processing to completely remove the involvement of manual notary.
ii. The image processing will also help in differentiating official documents and non official one.

REFERENCES

WEBSITES:

- www.stackoverflow.com/ethereum
- www.ethereum.org
- www.w3school.com/css
- www.simplifiedcoding.net/bootstrap
- www.blockchain.io

BIBLIOGRAPHY

- **Blockchain Revolution**: How the Technology Behind Bitcoin Is Changing Money, ... (By: Alex Tapscott and Don Tapscott)
- **Web Design Patterns**: Interaction Design Solutions for Developers

  (By: Greg Nudelman)
- **Angular 4 Book:** (By:Nathan Wu)
- **Html and Css:** Design and Build Websites (By Jon Duckett)